UNIVERSITY OF COPENHAGEN

MASTER THESIS

Multivariate contingent claims

Author:
Peter Pommergård LIND

Supervisor:
Dr. David SKOVMAND

A thesis submitted in fulfillment of the requirements
for the degree of Master Thesis in Actuarial Mathematics

October 9, 2020

Declaration of Authorship

I, Peter Pommergård LIND, declare that this thesis titled, "Multivariate contingent claims" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

"You were hired because you met expectations, you will be promoted if you can exceed them."

Saji Ijiyemi

UNIVERSITY OF COPENHAGEN

Abstract

Department of Mathematical Science
Science

Master Thesis in Actuarial Mathematics

Multivariate contingent claims

by Peter Pommergård LIND

The Thesis Abstract is written here (and usually kept to just this page). The page is kept centered vertically so can expand into the blank space above the title too...

Acknowledgements

I would first like to thank my thesis advisor Associate Professor David Skovmand of the department of Mathematical Science at University of Copenhagen. Prof. Skovmand consistently allowed this paper to be my own work, but steered me in the right the direction whenever he thought I needed it.

I would also like to thank Joakim Pagels and Emil Petersen for providing useful discussions in the process. A special thanks go to senior lecturer Mark Laplante at University of Wisconsin Madison for be such an inspiring lecturer and discovering my interest in finance.

Finally, I must express my very profound gratitude to my parents and to my family and friends for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

Peter Pommergård Lind

Contents

List of Figures

List of Tables

List of Abbreviations

ATM	**A**t **T**he **M**oney
B-S	**B**lack-**S**choles
BM	**B**rownian **M**otion
FPT1	**F**undamental **P**ricing **T**heorem **I**
FPT2	**F**undamental **P**ricing **T**heorem **II**
GBM	**G**eometric **B**rownian **M**otion
ITM	**I**n **T**he **M**oney
LIBOR	**L**ondon **I**nter**b**ank **O**ffered **R**ate
MLPs	**M**ulti**L**ayer **P**erceptron**s**
OTM	**O**ut **T**he **M**oney
RNVF	**R**isk **N**eutral **V**aluation **F**ormula
SDE	**S**tochastic **D**ifferential **E**quation
S-F	**S**elf-**F**inancing

List of Symbols

$\mathbf{A}$	matrix notation for matrix A
$\boldsymbol{a}$	vector notation for vector a
c	European call option price
p	European put option price
K	Strike price
T	Maturity in years
σ	Volatility of asset
C	American Call option price
P	American Put option price
S_0	Spot price
S_T	Stock price at maturity
$S_i(t)$	i'th stock price at time t
r	Continuous compounding risk-free yearly interest rate
$V^h(t)$	Value process
X	Simple Derivative
Φ	Contract function
W_t	Weiner process under martingale measure Q (synonym brownian motion)
$\bar{W}_t$	Weiner process under probabilty measure P
p_{ij}	Correlation coefficient between asset i and j
μ_i	drift of the continuous lognormal distribution
$F(t, S(t))$	pricing function of S(t) to time t
d	number of risky assets

For/Dedicated to/To my…

Chapter 1

Introduction

In recent years we have seen an increasing complexity of financial products, where big investment- and banks use a lot of money on financial engineerers in creating new innovative products. With the complexity a lot of challenges has risen in this field. Nevertheless the products can help to mitigate risk and leverage your portfolio. A recent example from the financial crisis in 2007 where credit default swap (CDS) almost led to AIGs bailout. A CDS is a derivative, where you insure your risk of losing money on some financial product. The strategy of writing CDS seemed like a good business for AIG as long there was a bull market, because they got good feeds for insuring credit. The CDS was the main reasions that AIG needed a bailout by the US government under the recent financial crisis. In hindsight they wrote to many CDS, hence AIG was too exposed for risk. A great understanding in the financial derivatives is important to understand your risks and ulitmately migate the damage of financial turmoil as Warren Buffett says derivatives is "Financial weapons of mass destruction" (page 15 (Buffett, 2002)). Eventhough Buffett is critical against derivatives he acknowlegde the usage of derivatives, because he owns derivatives in his portfolio. Derivatives gives the trader more options either to utilize arbitrage, speculate or hedge, but without care or knowledge about your book of derivative the outcome can be disastrous (Buffett, 2008).

The focus is on financial derivatives, where the prime examples will be plain vanilla stock options. We will start with the most basic derivatives European options and move toward more complex products like American options. The European option will be the reference point for our different nummerical approaches to American options, because the European option has a closed form solution (see proposition 3.2.1). When moving into more complex derivatives as American options the Black Scholes analytical framework breaks down, and this calls for numerical methods. We will take different numerical approaches for pricing and hedging, where the ultimate goal is to use machine learning for pricing and hedging. the techniques within the derivatives pricing industry are becoming more mathematical and rigourous with each passing year. In order to develop new exotic derivatives instruments, as well as price and hedge them, the financial industry has turned to academia. This has lead to the formation of mathematical finance research groups - academics who specialise in derivatives pricing models, risk analysis and quantitative trading. The goal is not to review continuous-time stochastic calculus but rather explore the applications and numerical procedures springing from the underlying theory.

Chapter 2

Arbitrage Theory In Continuous Time Finance

Arbitrage theory in continuous time finance is a field with a lot of technical details from probability theory, where we follow the style in (Hull, 2017) and (Björk, 2009) to focus on intuition without going into the whelm of technicalities and proofs. The focus on this chapter will to provide the basic tools and intuition for the arbitrage theory and lay the foundations for the computational finance methods. The key question is how to price derivative fair and hedge the risk imposed by the derivative. The thesis will mainly deal with the former, where the concepts of arbitrage and compleness will be important.

We start with introducing the financial markets and key concepts for building arbitrage free and complete market models (see section 2.1). Then we actually build a framework for finding "fair" prices, i.e. finding a complete model with absense of arbitrage (see section 2.2). Lastly we go into specific cases where either a closed-form solution exist or numerical methods is needed(section 2.3 and 2.4).

2.1 Financial Markets

In the financial markets there is a lot of players and different types of investments. The classical investments are bonds and stocks, where the big players in the markets are commercial banks, investment banks, insurance companies and pension funds. Besides the classical investments types derivatives gives additional options for investments. A derivative are a financial instrument depending on an underlying asset, where the dependency is specified in the contract. The options in this thesis will all be stock options, but the techniques developed can easily be extended to other types of derivatives. To find prices of derivatives in modelling we restrict our financial market to d risky assets $\boldsymbol{S}(t)$ and a bank account $S_0(t)$ as numeraire. The probability space $(\Omega, \mathcal{F}, P)$ with a filtration $\mathbb{F} = (\mathcal{F}_t)_{t\geq 0}$ is the fundamental for modelling stochastic processes describing asset prices and trading strategies, where in the thesis the probability space $(\Omega, \mathcal{F}, \mathbb{F}, P)$ will be implicit assumed. Intuitively the filtration $\mathcal{F}_t$ is the information observable to time t, where the filtration $\mathbb{F}^W$ is often used in the thesis and it is be generated by the wiener processes $(W_t)_{0\leq t\leq T}$. The bank account is assumed to be a strictly positive adapted process $S_0 = (S_0(t))_{t\geq 0}$ and $S_0(0) = 1$, where the d risky assets are modelled by a $\mathbb{R}^d$ adapted stochastic proces $\boldsymbol{S} = (\boldsymbol{S}(t))_{t\geq 0}$. The risky assets are in this thesis stocks by financial reasons the stocks are assumed positive $S_i(t) \geq 0$ P-a.s for all i and t. By using the bank account as numeraire i.e. dividing dividing the traded asset by the bank account

($\frac{S(t)}{S_0(t)}$), this amounts to working with *zero interest*. We assume that the our financial market is frictionless, which contains a lot of assumptions.

Assumption 2.1.1. *We assume following institutional facts:*

- *Short positions and fractional holding are allowed*
- *There are no bid-ask spread, i.e. selling price is equal to buying price*
- *There are no transactions costs of trading.*
- *The market is completely liquid, i.e. it is possible to buy/sell unlimited quantities on the market. You can borrow unlimited amount from the bank by selling short.*

(see p. 6 (Björk, 2009))

Besides the assumptions in (Björk, 2009) will we assume the market gives same uniform price for borrowing money and stocks are fixed stochastic processes exogenously and a priori given. All the assumptions is not realistic in real financial markets, but it gives a reasonable approximation. The assumptions can be modified but this complicates the mathematical convience, but the financial market is the key to price derivative in arbitrage teory.

2.1.1 Financial Derivatives

A financial derivation is a contract on some underlying assets, where the contract of can be constructed in many ways. We investigate stock options with different types of contracts, where we will mainly divide derivatives into two classes.

1. Simple stock derivatives (T-claims - european options)
2. Exotic stock derivatives (e.g. American options)

The first class are simple because you can only exercise them at maturity (time T) and we actually have a closed form solution for such options (section 3.2.1). The exotic derivatives are all kind of functions on the underlying assets, where you can e.g. have an option to exercise from inception to maturity (see section 2.4) or a contract on several underlying stocks (see section **??**).

Definition 2.1.1. European Call And Put Option: A European call option is an option where the owner of the option has the option to buy the underlying asset to price K at maturity. If the owner of the option chooses the buy the underlying asset, then the option is exercised. The contract function for the european call option:

$$\Phi(S(T)) = \max\{S(T) - K, 0\} \tag{2.1}$$

The put option is the right to sell the underlying asset to price K at maturity, hence the contract function for the european put option:

$$\Phi(S(T)) = \max\{K - S(T), 0\} \tag{2.2}$$

Where S(T) is the price of underlying asset at maturity and K is the agreed strike price.

The american option adds the feature to the option, that you can exercise at anytime beetween inception of the contract until maturity. This feature makes the american option an optimal stopping problem and the exercise gain for the american options is called the intrinsic value (section 2.4). For the american put option the payoff function at the stopping time is the same as for the european put at maturity.

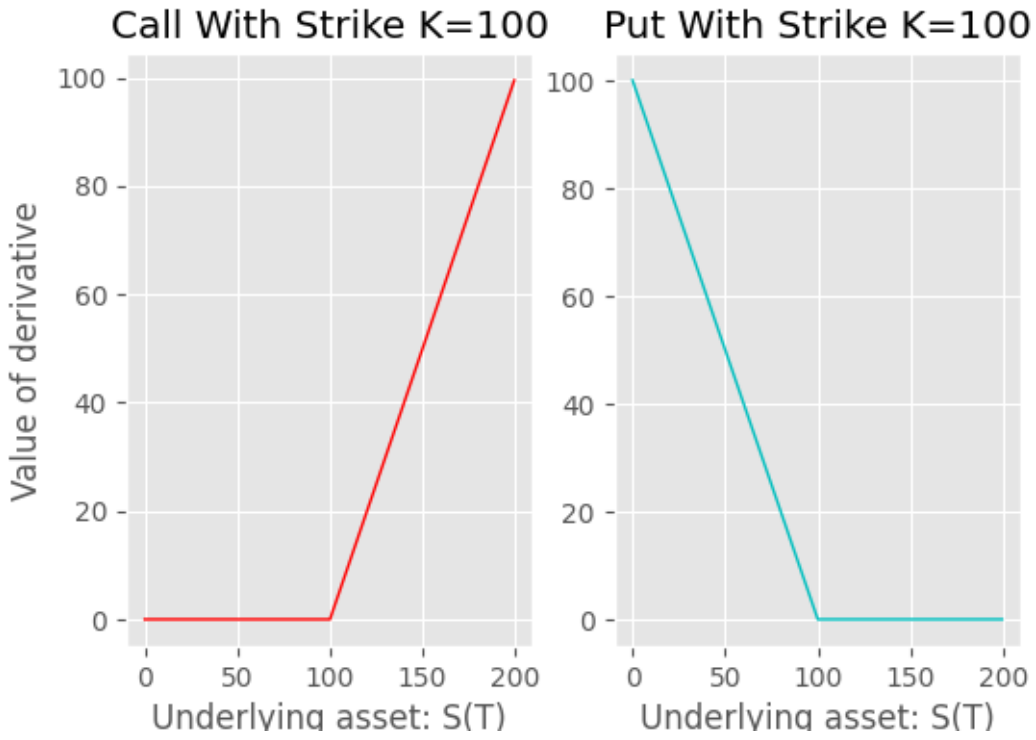

FIGURE 2.1: European options payoff at maturity with strike K=100

From the above illustration is it clear, that the owner of the option has limited downside, but the illustation does not take into account the initial price for the option. The profit and loss ($P\&L$) graph is also a common way of illustrating the payoff for an option, where you take the initial cost of buying the option into account. The fair price for such type of contract and other contracts will be the topic for this thesis, where the concepts of completeness and arbitrage will be central.

2.1.2 Self-financing Portfolio (Without Consumption)

Before being able to use the concepts of arbitrage and compleness, the construction of the portfolio from the financial market model will be important. The portfolio is the number of each assets of the market the owner of the portfolio holds. The value of the portfolio for a market model with the bank account and d stocks:

$$V^h(t) = \sum_{i=0}^{d} h_i(t) S_i(t) \tag{2.3}$$

V^h is called the value process and $h_i(t)$ is number of shares of type i during the period "$[t, t+dt)$". For the definition of arbitrage (Definition 2.5) we need to restrict

ourselfes to self-financing (S-F) portfolios. A self-financing portfolio h, is a portfolio h which doesn't get any external injection of money.

Definition 2.1.2. Self-financing portfolio A portfolio consisting of d+1 assets: h(t)=$(h_0(t), h_1(t), \ldots, h_d)$ is self-financing if:

$$dV^h(t) = \sum_{i=0}^{d} h_i(t) dS_i(t) \tag{2.4}$$

Where S_i is the $i'th$ asset in our portfolio, d+1 is the total number of assets in our market model and
$V^h(t) = \sum_{i=0}^{n} h_i(t) S_i(t)$

When dealing with discrete time finance the S-F portfolio is actually a budget restriction, this is important intuition for the continuous time version, because the continuous time version can be thought of the limit of the discrete version by letting step sizes in time tending to zero. To avoid pathological effects on the portfolio one often introduce the concept of an admissible portofolio:

Definition 2.1.3. a-admissible portfolio For $a \geq 0$, a portfolio h is called a-admissible if its value process $V^h(t)$ is uniform bounded from below by -a. A portfolio h is admissible if it is a-admissible for some $a \geq 0$.

The definition of a-admissible portofolio avoid situations as the doubling strategy known from gambling and imposes a limit to the debt arrangement. The important takeaway is that the S-F portfolio you only reallocate your assets through time within the portfolio.

2.1.3 Arbitrage

Arbitrage is the financial term for a "free lunch". An arbitage opportunity produces something out of nothing without risk, where the efficient market assumptions tells us in a well function market the "money pumps" cannot exist for long, because they would quickly be corrected by exploitation. In order to avoid making a "money machine" in our market, we want to price derivatives by not introducing arbitrage to the market.

Definition 2.1.4. Arbitrage: An arbitrage possibility on a financial market is an admissible self-financed portfolio h suct that

$$\begin{aligned} V^h(0) &= 0 \\ P(V^h(T) \geq 0) &= 1 \\ P(V^h(T) > 0) &> 1 \end{aligned} \tag{2.5}$$

The financial market $\boldsymbol{S}$ is called arbitrage-free if there exist no arbitrage opportunities. Sometimes $\boldsymbol{S}$ is said to satisfy (NA)
(p. 96 (Björk, 2009))

From the above definition we see that arbitrage is a natural financial requirement for a financial market model, because the investor in a arbitrage portfolio starts with 0 dollars, and without injecting any money, the investor is certain of not losing any money. In addition he has a positive probability by ending up with more than 0 at maturity. To price the derivatives fair in the model, the derivative should not introduce arbitrage to the market. There are different versions of the (NA) definition,

where for our purpose the above defition is sufficient. The *NA* is not only desirable for the market, we would also like to be able to replicate the payoff of the derivative with the other assets in the financial market model. If every derivative can be replicated in the market, then we have a complete market.

2.1.4 Complete Market And Hedging

Hedging is an important topic for risk management, because it tells you have to risk neutralize your exposure, i.e. a hedge is simply a risk neutralization action in order to minimize the overall risk. In the definition below, we define a hedge for an simply T-claim.

Definition 2.1.5. Hedging and completeness for T-claim: A T-claim X can be hedged, if there exist a self-financing portfolio h such that:

- $V^h(T) = X$ P-a.s.

I.e. h is an hedge portfolio for X if it is guaranteed to pay in all circumstances an amount identical to the payout of X.
The market is complete, if every derivative is hedgeable. (p. 192 (Björk, 2009))

By introducing the basic concepts for how to price fair and protect ourselves against financial risk, we will in next section focus on building the financial market model.

2.2 Multidimensional Models

There is two main method for deriving arbitrage free and complete markets. The classical approach is the delta hedging approach (Black and Scholes, 1973) and (Cox and Stephen Ross, 1979)). The more advanced mathematical approach is the martingale approach (Björk, 2009). In this section we will focus on the martingale approach and show that delta hedging approach is a special case of the more general martingale theory. For the martingale approach the First and Second Fundamental Theorems of Mathematical Finance will be the key for obtaining a fair market. Besides the model assumptions will we also assumes the financial market assumptions in section 2.1.

2.2.1 Model Assumptions

Let us consider a filtered probability space $(\Omega, \mathcal{F}, P, \mathcal{F}_t^{\bar{W}})$. Note the assumption that filtration is generated from the Wiener process, so the $\bar{W}$ is the only random source and we assume $\bar{W}_i$ is k-dimensional. I.e. we assume that we are in a Wiener world, where all processes are Wiener driven. A priori we assume a market $(B(t), S_1(t), S_2(t), \ldots, S_d(t))$, where $S_i(t)_{i=1,2,\ldots,d}$ are d risky assets and B(t) is the risk free asset. By assumptions their dynamics are given by:

$$d\boldsymbol{S}(t) = D[\boldsymbol{S}(t)]\boldsymbol{\alpha}(t)dt + D[S(t)]\boldsymbol{\sigma}(t)d\bar{\boldsymbol{W}}(t) \tag{2.6}$$

$$dB(t) = r(t)B(t)dt \tag{2.7}$$

We assume α_i, σ_{ij} and the short rate $r(t)$ are adapted processes. The evolution of the stocks are described by a geometric brownian motion (GBM) which has a solution to

the SDE. The randomness comes from the brownian motion (BM) in the GBM, which has wildly tracjectories. The function $t \mapsto W_t(\omega)$ from $[0, \infty)$ to $\mathbb{R}$ is continuosus, but nowhere differentiable. Furthermore has the BM nonzero quadratic variation and infinite variation, but it also process well-behaved property e.g. the BM is a Lévy process. For illustation figure 2.2 we plottet three approximations to three sample paths of the GBM with initial value $S_i(0) = 36$.

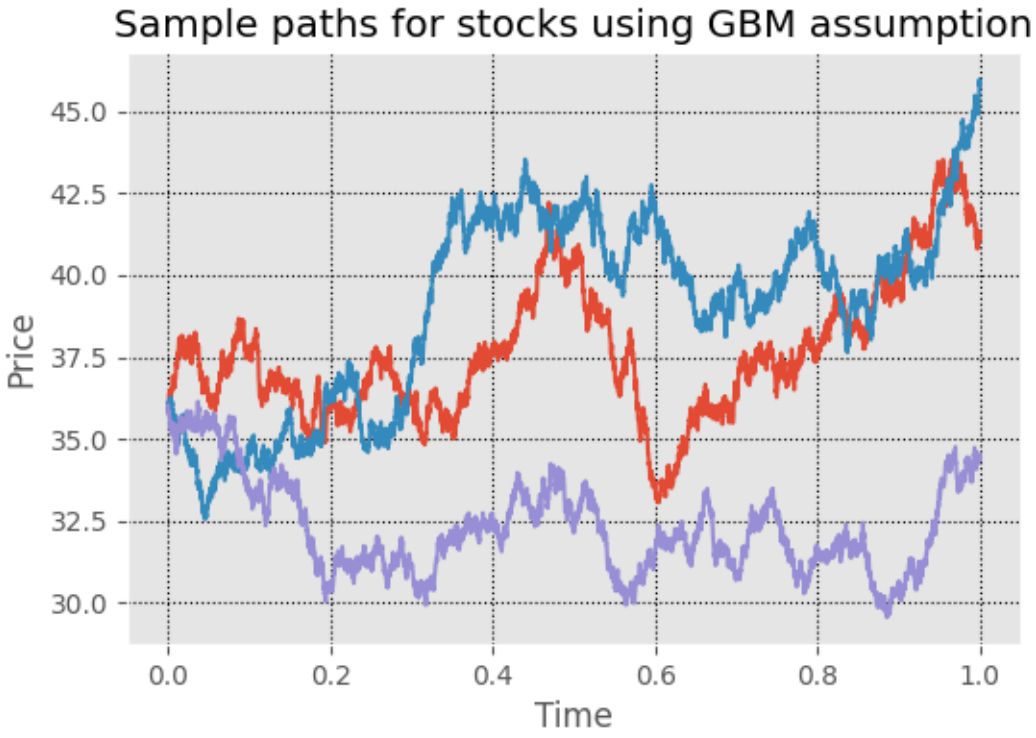

FIGURE 2.2: Three sample paths for stocks under GBM assumptions, where the spot is \$36, σ=0.2 and α=0.06

The tool for handling BM is stochastic calculus in continuous time, because the standard calculus will not work for the wildly behaved BM. In the representation of the GBM, we used vector and matrix notation for the GBM process. The stock vector is d dimensional and the wiener process vector is k dimensional. The volatility matrix is given by $\sigma = \{\sigma_{ij}(t)\}_{i=1,\dots,d,j=1,\dots,k}$ and the local mean of rate of return vector is $\alpha = (\alpha_1(t), \alpha_2(t), \dots, \alpha_d(t))^T$. D(x) denotes a diagonal matrix with vector x as its diagonal and the Wiener processes covariance is $Cov(dW_i(t)dW_j(t)) = \rho_{ij}dt$ where $\rho_{i,i} = 1$

2.2.2 Arbitrage Free Model

The first problem we are faced with in arbitrage theory is to create a model with no arbitrage opportunities. The First Fundamental Theorem tells us how to not introduce arbitrage to our market model.

Theorem 2.2.1. ***First Fundamental Pricing Theorem of Mathematical Finance(FFT1):*** *The market model is free of arbitrage if and only if there exist a equivalent martingale measure, i.e. a measure $Q \sim P$ s.t. the processes:*

$$\frac{S_0(t)}{S_0(t)}, \frac{S_1(t)}{S_0(t)}, \cdots, \frac{S_d(t)}{S_0(t)}$$

are (local)martingales under Q. (see p. 154 (Björk, 2009))

From the FFT1 with the bank account B(t) as numeraire, we have:

Proposition 2.2.1. *We assume that $B(t) = S_0(t)$ is our numeraire and all the processes are Weiner driven, then a equivalent measure $Q \sim P$ is martingale measure if and only if all assets $(B(t), S_1(t), \ldots, S_d(t))$ have the short rate as their local rates of return, i.e.*

$$dS_i(t) = S_i(t)r(t)dt + S_i(t)\sigma_i(t)dW^Q(t) \tag{2.8}$$

(see p. 154 (Björk, 2009))

So to not introduce arbitrage to the model for the financial market, we need to ensure the Q-dynamics of S is:

$$dS(t) = D[S(t)]r(t)dt + D[S(t)]\sigma(t)d\bar{W}(t) \tag{2.9}$$

The tool to obtain the dynamics in eq. (2.9) is Girsanov theorem (see A.0.2). Girsanov theorem is a continuous measure transformation, where in our model we want to transform the dynamics given with the objective probability measure P to an equivalent martingale measure Q (i.e. the martingale measure chosen by the market). By suitable chooses of the likelihood process L and setting $dQ = L(T)dP$, then with Girsanov theorem:

$$d\bar{W}(t) = \phi(t)dt + dW(t)$$

When applying to eq. (2.6):

$$dS(t) = D[S(t)](\alpha(t) + \sigma(t)\phi(t))dt + D[S(t)]\sigma(t)d\bar{W}(t)$$

Going back to the FFT1 and the proposition, we know that Q is martingale measure if and only if:

$$\boldsymbol{\alpha}(t) + \boldsymbol{\sigma}(t)\boldsymbol{\phi}(t) = \mathbf{r}(t) \quad \textit{holds with probability } 1 \textit{ for each } t \tag{2.10}$$

We disregard pathological models when doing so the term generically arbitrage free will be used.

Definition 2.2.1. Generically arbitrage free: The model in this section is said to be generically arbitrage free if it is arbitrage free for every (sufficiently integrable) choice of $\boldsymbol{\alpha}$. (p. 198 (Björk, 2009))

Furthermore we assume enough integrability and we have the following useful result:

Proposition 2.2.2. *Disregarding integrability problems the model is generically arbitrage free if and only if, for each $t \leq T$ and P-a.s. the mapping: $\sigma(t) : \mathbb{R}^k \to \mathbb{R}^n$ is surjective, i.e. if and only if the volatility matrix $\sigma(t)$ has rank n. (see p. 198 (Björk, 2009))*

We note that in order not to have arbitrage in our model, we need $k \geq n$, i.e. have at least as many random sources as number of risky assets.

2.2.3 Complete model

Second Fundamental Pricing Theorem is key to obtain a complete market model, i.e. a market model where every claim can be hedged.

Theorem 2.2.2. ***Second Fundamental Pricing Theorem of Mathematical Finance(FFT2):*** *Assuming absence of arbitrage, the market model is complete if and only if the martingale measure Q is unique. (see p. 155 (Björk, 2009))*

Hence in our Wiener world we have a unique martingale measure if eq. 2.10 has a unique solution.

Proposition 2.2.3. *Assume that the model is generically arbitrage free and that the filtration is defined by:*

$$\mathcal{F}_t = \mathcal{F}_t^{\bar{W}}$$

Then disregarding integrability problems, the model is complete if and only if k=n and the volatility matrix $\sigma(t)$ is invertible P-a.s. for each $t \leq T$ (see p. 200 (Björk, 2009))

2.2.4 Pricing and connection to classical approach

The pricing formula for arbitrage free market model is the risk neutral valuation formula:

Proposition 2.2.4. ***Risk Neutral Valuation Formula*** *To avoid arbitrage, $\mathcal{X}$ must be priced according to the formula:*

$$\Pi(t;\mathcal{X}) = S_0(t)E^Q[\frac{\mathcal{X}}{S_0(T)}|\mathcal{F}_t] \tag{2.11}$$

Note if we choose our numeraire $S_0(t) = B(t)$ then

$$\Pi(t;\mathcal{X}) = E^Q[\exp(-\int_t^T r(s)ds)\mathcal{X}|\mathcal{F}_t] \tag{2.12}$$

(see p. 155 (Björk, 2009))

Proposition 2.2.4 will raise the question if there is more than one fair price for the derivative. The answer is found in FTT2, the market is complete if and only if the measure Q is unique. Intuitively it means that if you can replicate the derivative with a portfolio, then this hedge should only earn risk free interest rate.

The classical approach in (Black and Scholes, 1973) to arbitrage free and complete market models is based on a Markovian model assumption. For the model to have markovian property, we assume k=n and the probability space is $(\Omega, \mathcal{F}, P, \mathcal{F}_t^{\bar{W}_t})$. Furthermore we assume α and σ are deterministic and constant over time. σ is also assumed invertible. Under these more restrictive assumptions the risk neutral valuation formula for a simple T-claim is given by

$$\exp(-r(T-t))E^Q[\mathcal{X}|S(t)] \tag{2.13}$$

The Markov property implies that the price only depend on the current state of S, and then applying Kolmogorov backward equation on eg. 2.13, we obtain the BS-PDE for the pricing function F(t,S(t))=$\Pi(t;\mathcal{X})$.

Theorem 2.2.3. ***Black Scholes PDE:*** *Consider the contract $\mathcal{X} = \Phi(\mathbf{S}(T))$. In order not to introduce arbitrage to the market, the pricing function $F(t,s)$ must solve the boundary value problem.*

$$\begin{aligned} F_t(t,s) + \sum_{i=1}^{n} rs_i F_i(t,s) + \frac{1}{2} tr\{\sigma^* D[S] F_{ss} D[S]\sigma\} - rF(t,s) = 0 \\ F(T,s) = \Phi(s) \end{aligned} \tag{2.14}$$

(see p. 203 (Björk, 2009))

2.3 Classical Black-Scholes Formulas

We will not do the classical delta hedging approach in (Black and Scholes, 1973). Instead we use the general multidimensional martingale approach to derive the essential formulas for pricing. To derive a closed-form solution to the European call and put option, we concentrate at a special case of the multidimensional framework, where we only have the risk free asset and one risky asset in the financial market model. We further restrict ourselves to:

Assumption 2.3.1. ***Black-Scholes assumptions****: We assume following ideal conditions in addition to (2.1.1):*

- *The short-term interest rate is known and is constant through time*
- *The stock price follows a Geometric Brownian Motion. The σ is constant.*
- *The stock pays no dividends or other distributions.*
- *The option is a simple option ("European").*

(p. 640 (Black and Scholes, 1973))

We assume the underlying stock follows a geometric brownian motion: $dS(t) = \alpha S dt + \sigma S dW_t$ where the solution to the SDE is given as

$$S(t) = S(0) \cdot \exp\left((\alpha - \frac{1}{2}\sigma^2)t + \sigma W(t) \right) \tag{2.15}$$

Where α is the local mean rate of return and σ is the volatility of S are constants. By above assumptions we are in a Markovian model, and we know that the Black Scholes PDE in this setup (see eq. 2.2.3) from the risk neutral valuation formula. The solution of the SDE of S under Q dynamics is:

$$S(t) = S(0) \cdot \exp\left((r - \frac{1}{2}\sigma^2)t + \sigma W(t) \right)$$

By the simple option assumption the Kolmogorov backward equation for the RNVF gives the BS-PDE. The PDE can then be solved analytically and then we arrive at a closed form solution for european call and put options. The european call and put option can also be derived directly with the RNVF.

Proposition 2.3.1. ***Black-Scholes formula for call option:*** *The price of a European call option with strike K and maturity T is given by the formula $\Pi(t) = F(t, S(t))$, where*

$$F(t,s) = s \cdot N(d_1(t,s)) - e^{-r(T-t)} \cdot K \cdot N(d_2(t,s))$$

N is the cumulative distribution function of a standard normal distribution $\mathcal{N}(0,1)$ *and*

$$d_1(t,s) = \frac{1}{\sigma \cdot \sqrt{T-t}} \cdot \left(\ln(\frac{s}{K}) + (r + \frac{1}{2}\sigma^2)(T-t) \right)$$
$$d_2(t,s) = d_1(s,t) - \sigma\sqrt{T-t}$$

(see p. 105 (Björk, 2009))

We provided only the price for the european call option, but the european put price can easily be obtained by applying the put-call-parity for european options and vice virsa .

Proposition 2.3.2. ***Put-call parity:*** *Assume the call and put option has same strike price and time to maturity.*

$$p(t,s) = K \cdot \exp(-r(T-t)) + c(t,s) - s$$

(see p. 126 (Björk, 2009))

The aim for this thesis is to price american put options, but the european option provide a reference price in a closed form format. The put-call-parity holds only for european options, where for the american option there is a bound on the difference in price:

$$S_0 - K \leq C - P \leq S_0 - K \cdot e^{-rT}$$

The above formula for the European call option is actually the same for an American call option, but is not true for an American put option or for call options with underlying stock paying dividends. The result for the American call option was shown by Merton (Merton, 1973), that the intrinsic value is never greater than the worth of the option given by the risk-neutral valuation formula. In section 2.4 we will show a martingale approach to prove the value of a European and American call coincides when the underlying is a non-dividend paying stock.

2.4 American Options And Optimal Stopping

The American options adds additional complexity to the pricing problem, because compared to the European option the American option can be exercised at any time from inception to maturity. The exercise feature of the american option makes a optimal stopping problem for american put option. We will assume a finite horizon, because all the derivative in this thesis will be prices in that time frame. The arbitrage free price for an american put option with finite horizon is:

$$V(t,s) = \sup_{0 \leq \tau \leq T-t} E_{t,s}[\exp(-r\tau)(K - S(t+\tau))^+] \tag{2.16}$$

Equation 2.16 gives a describtion how to price the american put option, the V(t,s) is the value of the option to time t and given $S(t) = s$. τ is a stopping time of the GBM.

Definition 2.4.1. Stopping time: A random variable $\tau : \Omega \to [0, \infty]$ is called a Markov time if $(\tau \leq t) \in \mathcal{F}_t$ for all $t \geq 0$. A Markov time is called a stopping time if $\tau < \infty$ P-a.s. (p. 27 (Shiryaev and Peskir, 2006))

The value of the options is given by exercising the option at the optimal stopping time, hence the problem is a optimal stopping problem. TODO!

The main problem with American options is to find a optimal stopping time, i.e.

$$\max_{0\leq\tau\leq T}\{E[\Phi(\tau, X_\tau)]\} \tag{2.17}$$

Where τ is a stopping time (see definition A.0.1). We assume satisfied integrability condition on a finite interval $[0, T]$:

$$\sup_{0\leq\tau\leq T}\{E[\Phi(\tau, X_\tau)]\} < \infty$$

and assume a diffusion setting:

$$dX_t = \mu(t, X(t))dt + \sigma(t, X(t))dW(t)$$

To find the optimal stopping time we introduce the optimal value function V(t,X(t)).

Definition 2.4.2. Optimal value function For fixed $(t, x) \in [0, T]X\mathbb{R}$, and each stopping time τ with $\tau \geq t$ the optimal value function $V(t, x)$ is defined by

$$V(t, x) = \sup_{t\leq\tau\leq T}\{E[\Phi(\tau, X_\tau)]\} \tag{2.18}$$

A stopping time which realizes supremum for V is called optimal and be denoted $\hat{\tau}_{tx}$. (see page 341 (Björk, 2009))

By using a dynamic programming argument with three strategies:

- Use optimal stopping strategy $\hat{\tau}_t$
- Stop immediately
- Wait one time-step h and then use optimal stopping strategy $\hat{\tau}_{t+h}$

Jumping over some argument and like in this section assuming "enough regularity", we arrive at two important propositions for numerically evaluating American options.

Proposition 2.4.1. ***variational inequalities*** *Given enough regularity, the optimal value function is characterized by the following relations:*

$$\begin{aligned} V(T, x) &= \Phi(T, x) \\ V(t, x) &\geq \Phi(t, x) \quad \forall(t, x) \\ \left(\frac{\partial}{\partial t} + \mathbb{A}\right)V(t, x) &\leq 0 \quad \forall(t, x) \\ \max\left\{V(t, x) - \Phi(t, x), \left(\frac{\partial}{\partial t} + \mathbb{A}\right)V(t, x)\right\} &= 0 \quad \forall(t, x) \end{aligned} \tag{2.19}$$

Where $\mathbb{A}$ is the Itô operator:

$$\mathbb{A}f(t, x) = \mu(t, x)\frac{\partial f(t, x)}{\partial x} + \frac{1}{2}\sigma^2(t, x)\frac{\partial^2 f(t, x)}{\partial x^2}$$

(see p. 344 (Björk, 2009))

Proposition 2.4.2. ***Free boundary value problem*** *Assuming enough regularity, the optimal value function satisfies the following parabolic equation*

$$\begin{cases} \frac{\partial V(t,x)}{\partial t}(t, x) + \mu(t, x)\frac{\partial V(t,x)}{\partial x}(t, x) + \frac{1}{2}\sigma^2(t, x)\frac{\partial^2 V(t,x)}{\partial x^2} = 0 & (t, x) \in C \\ V(t, x) = \Phi(t, x) & (t, x) \in \partial C \end{cases}$$

Where C is the continuation region defined by:

$$C = \{(t,x) : V(t,x) > \Phi(t,x)\}$$

(see p. 343-344 (Björk, 2009))

We will see in the American put section why these two propositions are useful.

2.4.1 American Call Without Dividends

The American call options is a special case, because the optimal stopping time is always at the options maturity. With martingale machinery it means the value-process is a submartingale, which mean the $\hat{\tau} = T$.

The optimal stopping problem is:

$$\sup_{0 \leq \tau \leq T} \{E[\exp(-r\tau)\max\{S_\tau - K, 0\}]\}$$

Hence we want to maximize the expectation of the process:

$$\max\{\exp(-rt)S_t - \exp(-rt)K, 0\}$$

From the theory developed we know that $\exp(r \cdot t) \cdot S_t$ is a Q-martingale and $\exp(r \cdot t) \cdot K$ is a deterministic decreasing function hence a supermartingale. Then $\exp(r \cdot t) \cdot S_t - \exp(r \cdot t) \cdot K$ is a submartingale. Applying the function max which is a convex and increasing functionther on a submartingale is still a submartingale. Hence the optimal stopping time is $\hat{\tau} = T$.

2.4.2 American Put

For the American put the optimal stopping problem is:

$$\max_{0 \leq \tau \leq T} \{E[\exp(-r\tau)\max\{K - S_\tau, 0\}]\}$$

There is no analytical formula for American put, hence numerical procedures are required. For practical use there are three strategies to find the fair price for the option:

- Solve the free boundary free problem numerically (see 2.4.2)
- Solve the variational inequalities numerically (see 2.4.1)
- Approximate the Black-Scholes model by a binomial model and compute the exact binomial American put price.

(Björk, 2009)

We will in the following chapters try to valuate with both the binomial model and solving the variational inequalities.

Chapter 3

Classical numerical results and Benchmarks

By last section we saw the American put was an example of an option that required numerical procedures to be priced fair. The American put is far from the only example of a derivative without a closed-form solution. In this chapter the two first sections deals with pricing American put option with 1 underlying risky asset, where in the last section we try to price options with several underlying risky assets.

The two first sections is two classical valuing algorithms in computational finance the Cox-Ross-Rubinstein (CRR) binomial model (Cox and Stephen Ross, 1979) and the Least Square Monte Carlo (LSM (Longstaff and Schwartz, 2001)) approach with one underlying asset. The binomial model is an example of a strategy to approximate the B-S model and the LSM is a method trying to solve the variational inequalities. We could also have chosen to solve the free boundary problem with implicit finite difference, but we chose to focus on the two other numerical procedures. The final section in this chapter will be trying to value exotic options with several underlying assets. Here we will extend the binomial pricing model to multidimensional ((Ekvall, 1996) and (Boyle, Evnine, and Gibbs, 1989)) and provide some closed form solutions ((Johnson, 1987) and (Ouwehand, 2006)). Therefore the chapter have two purposes to gain insight into valuation for exotic options and provide some benchmarks for the Neural Network in the coming chapters.

3.1 Cox Ross Rubenstein Model

The classical binomial model presented in this section inspired by (Cox and Stephen Ross, 1979; Hull, 2017; Björk, 2009) will be used for pricing an american put stock option and to build the foundation for the multidimensional binomial model (Boyle, Evnine, and Gibbs, 1989). The Binomial model provides an intuitive and easy implementable model for valuing american and european options. The binomial model comes handy, when no analytical model exists e.g. for an american put option. The Binomial model also has its limitations, because it is not suited for valuing path dependent options or options with a lot of several underlying factors. The key difference on the Binomial model and the other numerical procedures is that the binomial model is build on a discrete framework.

We work with the financial market $(\Omega, \mathcal{F}, \mathbb{F}, P, S_0, S_1)$, where the filtration is generated by S_1, $\mathbb{F} = \sigma(\mathcal{F}_k)_{k=0,1,\dots,T}$ and the sigma algebra is chosen to be $\mathcal{F} = \mathcal{F}_T$. It is well known from discrete arbitrage teory, that the binomial market model with two assets, where $u > r > d > -1$ is complete and arbitrage free model. The u,d and

r describes the evolution of the discrete stochastic process for the stock and the free interest rate on the bank account.

$$S_0(k) = S_0(0) \cdot \exp(\Delta t \cdot r \cdot k) \quad \textit{where } S_0(0) = 1$$
$$S_1(k) = S_1(0) \prod_{j=1}^{k} Y_j \quad \textit{where } Y_1, Y_2, \ldots, Y_k \textit{ are i.i.d. and } S_1(0) > 0$$

We assume that the $Y_i =$

$$\begin{cases} u & \textit{with probability } p \\ d & \textit{with probability } (1-p) \end{cases}$$

Remember that in arbitrage teory we work with the market expectation, hence we want the equivalent martingale measure. The risk neutral valuation formula holds also in discrete time:

Theorem 3.1.1. ***Risk-neutral valuation formula in discrete time.*** *Assume there exists a risk free asset. Then the market is arbitrage free if and only if there exists a risk neutral measure $Q \sim P$ s.t.*

$$s = \exp(-r\Delta t) \cdot E^Q[S(t+\Delta t)|S(t) = s] \tag{3.1}$$

Where Δt is a single time-step.

From the above theorem, we can calculate the equivalent martingale measure:

$$q = \frac{e^{\Delta t} - d}{u - d}$$

The martingale measure q is unique in the binomial model, because it is complete. We have seen that the binomial model with the market $(S_0(k), S_1(k))$ is a arbitrage free and complete model, hence we are left with the question how to introduce a derivative with payoff H. The binomial model gives a recursice formula, because the filtration $\mathbb{F}$ has the structure of a binary tree. I.e. the one-step transition probabilities are the same in each node throughout the tree, because of the i.i.d. assumption.

$$\begin{aligned} V(k-1) = \exp(-r\Delta t)(qV^u(k) + (1-q)V^d(k)) \\ \textit{with terminal condition } V(T) = H \end{aligned} \tag{3.2}$$

The path-independence payoff for the american put makes the tree recombining, so there is only T+1 terminal nodes at maturity. If the derivative was path-dependent e.g. an asian option, then we have a non-recombinin tree and 2^N terminal nodes. This is a computational inefficient, which explains that the binomial model should not be used for path-depending derivatives. The problem with a derivative with several underlying e.g. a basket option is also the increasing number of nodes, because now you have 2^d possible one-step transitions.

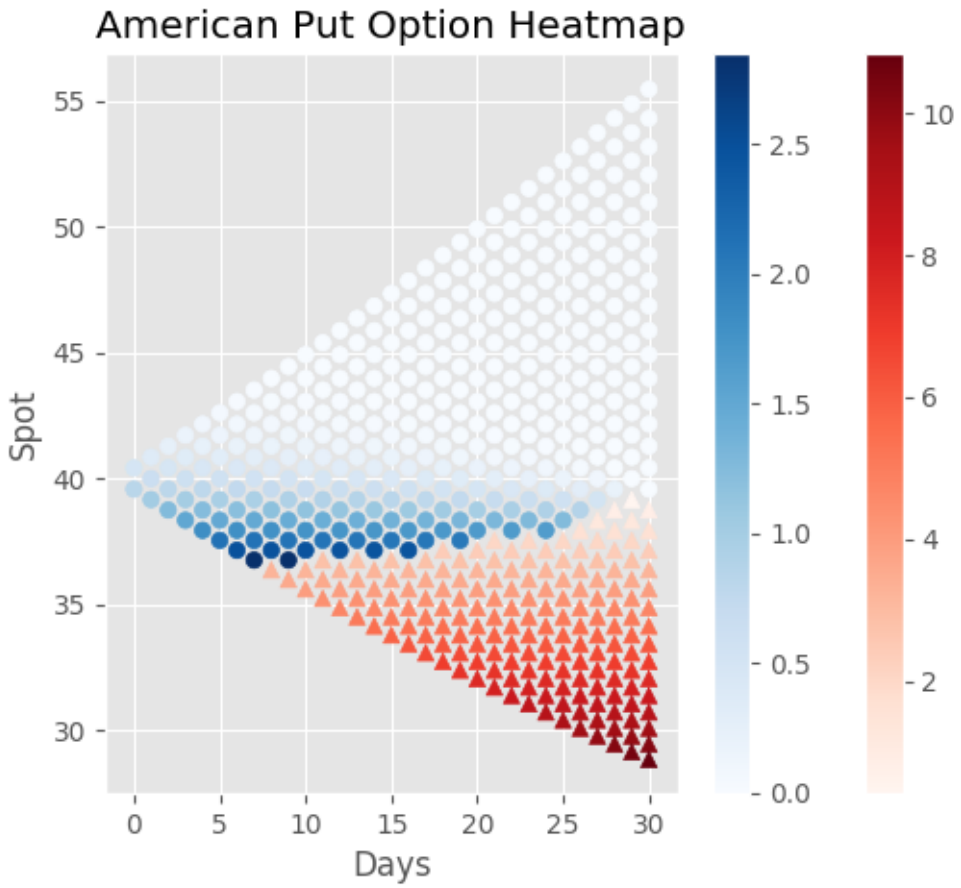

FIGURE 3.1: A valuation tree of an american put option price based on the binomial model, where the color indicate the value and the dots are marking the continuation nodes. The parameters are S(0)=40, N=30, $\Delta t = 1$ day, K=40 and u=1.0106

To value an American put option, we lay out all the possible path of the stock based on $S_1(0), \sigma$ and T. First we need to construct the tree, then afterwards work backwards in the tree for valuation. Figure 3.1 is an example of a constructed tree, where the value of the option is also included by color. To construct the tree we need to specify the number of equidistant time-steps Δt ($\Delta t = \frac{T}{N}$ *where* $N = No.\ of\ steps$) for the tree, where for each step we add another possible value for the stock. We only add 1 more possibility for each time-step because the tree recombines. The d and u is chosen s.t. they match volatility. So we choose:

$$u = \exp(\sigma\sqrt{\Delta t}) \quad d = \exp(-\sigma\sqrt{\Delta t})$$

For valuing an American put option, we value the exercise value at maturity (time T) for all possible outcomes for the stock. Then we use backward induction where we compare the intrinsic value with the conditional expectation, where we choose the maximum of these two. If we denote $V_j(k)$ as the value of the option to time $k \cdot \Delta t$ and j is the node at that time. For an put option we have at maturity:

$$V_j(k) = \max(K - S_1(0) \cdot u^j \cdot d^{N-j}, 0) \quad for\ j = 0, 1, \ldots, N\ and\ k = 0, 1, \ldots, T$$

At each node we compare the intrinsic value with the continuation value, where the continuation value is given for each node by the two possible one-step transitions:

$$V_j(k) = \exp(-r\Delta t)(q \cdot V_{j+1}(k+1) + (1-q) \cdot V_j(k+1)) \quad for\ j = 0, 1, \ldots, N$$

Remember from above that the binomial model has a recursive structure, hence use backward induction to value the option:

$$V_j(k) = \max(K - S_1(0) \cdot u^j \cdot d^{N-j}, \exp(-r\Delta t)(q \cdot V_{j+1}(k+1) + (1-q) \cdot V_j(k+1))) \quad for\ j = 0, 1, \ldots, N$$

The first term in the max function is the intrinsic value and the second term is the condition expectation by RNVF. The comparison will be applied for every node in each time-step Δt and all the way back in time to the initialization date. By this procedure we get present value of the American option. One design decision is to choose number of time-steps considering a trade-off between computational efficientcy and accurary. The precision for the algorithm increases with the number of steps and the option value stabilizes for increasing number of steps (see Figure 3.2)

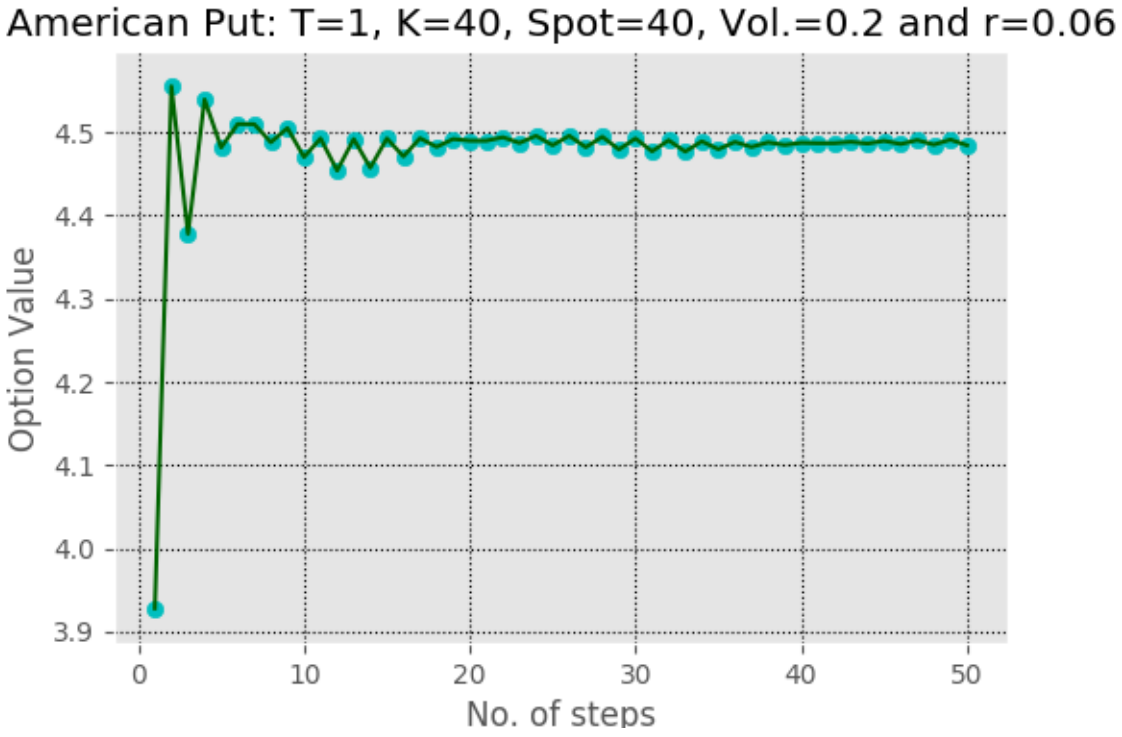

FIGURE 3.2: Price for a american put option based on the binomial model, where the independent variable is the number of time-steps. Vol. is an abbreviation for volatility.

Figure 3.2 illustrates that around 40 steps the option value stabilizes for an option with 1 year to maturity.

The central concepts arbitrage and completeness from continuous time also work in the discrete time setup. The paper (Cox and Stephen Ross, 1979) which introduced the binomial model to option pricing came after the Black-Scholes model described in section 2 (Black and Scholes, 1973). The main reason for developing a model in

discrete time, is that the discrete time approach gives a simplified model in terms of the mathematics and highlights the essential concepts in arbitrage theory. You can argue that the simpler mathematics in this model makes the binomial model more instructive and clear. Besides being easier to understand for non-mathematician it works nicely with other options than the european options like american options.

Even though we assume the stock price moves at discrete time instead in continuous time it can actually be shown for a European Option that if the number of time-steps in the tree approaches infinity. The Binomial model will then converge to the continuous time closed form solution for a European option (Cox and Stephen Ross, 1979) (Hull, 2017). Hence the binomial pricing model will be equivalent with the continuous time analytical pricing model derived by Fischer Black and Myron Scholes in the limit for European options (Cox and Stephen Ross, 1979).

3.2 Least Square Monte Carlo Method

The classical result in this section is of a different nature, because it is based on simulation and the linear model. This section will go through the valuation method assuming 1 underlying risky asset. To price a american put options, we are faced with the optimal stopping problem:

$$V_0 = \sup_{\tau \in \mathcal{T}(0,\dots,T)} E\{e^{-r\tau} \cdot \max\{K - S_{tau}, 0\}\} \tag{3.3}$$

where V_0 is the price and $\mathcal{T}(0,\dots,T)$ is a class of all $(0,\dots,T)$-valued stopping times. The stock values are modelled via black scholes theory (see path for GBM figure 2.2), hence the simulated evolution for the stock under the risk neutral valuation is given by:

$$S(t) = S(0) \cdot \exp\left((r - \frac{1}{2}\sigma^2)t + \sigma W(t)\right)$$

The computer is discrete by nature, so we approximate the american put option with a bermudan put option. The K time points chosen between inception and maturity are equidistant time steps and by chosen K sufficient large the bermuda option approximate the american option. To solve (3.3) backward induction is applied, where the idea like in the binomial case is to start at maturity and work backward in time.

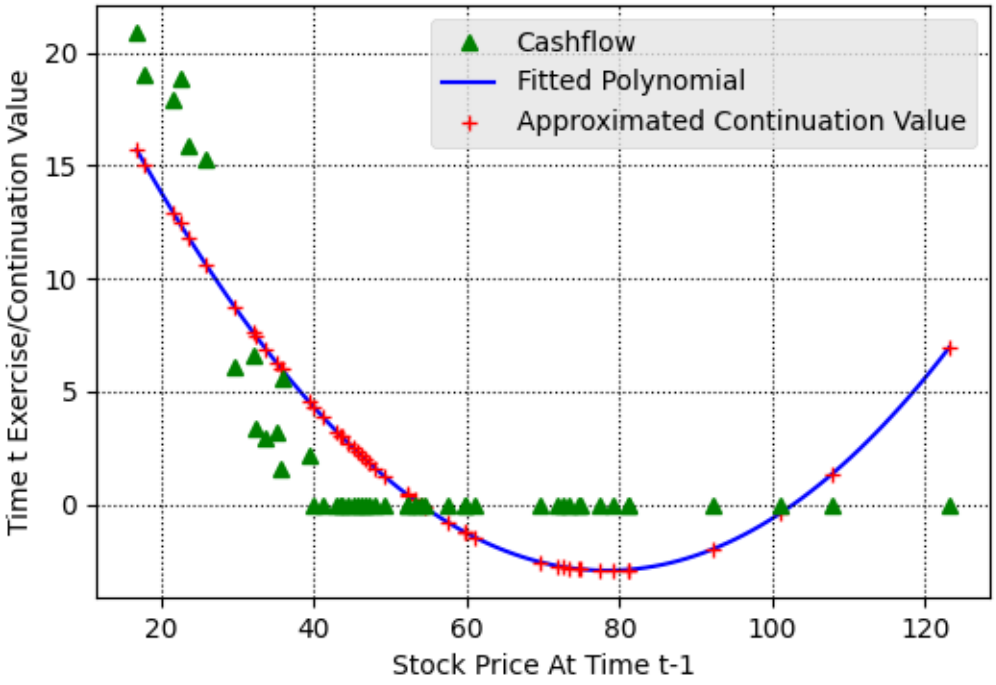

FIGURE 3.3: By zooming in on a specific point of time in backward induction approach, we see how the algorithm regress the continuation value

In our setting we regress the expected payoff by continuation of the contract and compare it to the intrinsic value. The dependent variable in the regression is the expected value of continuation and the independent variables is a set of orthogonal basis functions in $L^2(\Omega, \mathcal{F}, Q)$ of the simulated paths. Typical choices for basis functions could be weighted Laguerre -, Hermit -, and Jacob polynomials. This kind of regression is a nonlinear expansion of the linear model.

3.2.1 LSM method for an American put

We want to valuate an American put option with a stock as underlying asset. We take the same assumptions as in Chapter 2 (see assumption 2.3.1) except the option is an American option. Hence in order to simulate the paths of the stock, we simulate from a GBM: $dS(t) = rSdt + \sigma SdW_t$ where σ and r are constant (see solution to SDE equation 2.3). We simulate 100.000 paths for the stock. Like in the binomial model, we work backward to decide the optimal stopping time. The computer is discrete, hence we simulate the stock path as an Bermudan option, where we have 50 time-steps per year. I.e. we approximate the American option with a Bermudan option on same underlying.

At maturity the cash flow from the option is the same as for an European put option, hence the cash flow from each path is $C(\omega, T; T, T) = max(K - S_T, 0)$. We use the notation $C(\omega, s; t, T)$ denote the path of cash flows generated by the option condition on the option not being exercised before t and the option holder follow the optimal stopping strategy for all s, $t < s \leq T$. (inspired by (Longstaff and Schwartz,

2001) p. 121). The continuation value is given by:

$$F(\omega; t_k) = E^Q[\sum_{j=k+1}^{K} \exp(-\int_{t_k}^{t_j} r(\omega, s)ds)C(\omega, t_j; t_k, T)|\mathcal{F}_{t_k}] \tag{3.4}$$

where $r(\omega, t)$ is risk free interest rate, and the $\mathcal{F}_{t_k}$ is the filtration at time t_k.

We get the optimal stopping strategy by comparing thr continuation value with the intrinsic value at each time step. By working backward in time until the initialization of the option, we have approximated the the optimal stopping times and the cash flows associated with exercising at the optimal stopping times (see figure 3.4).

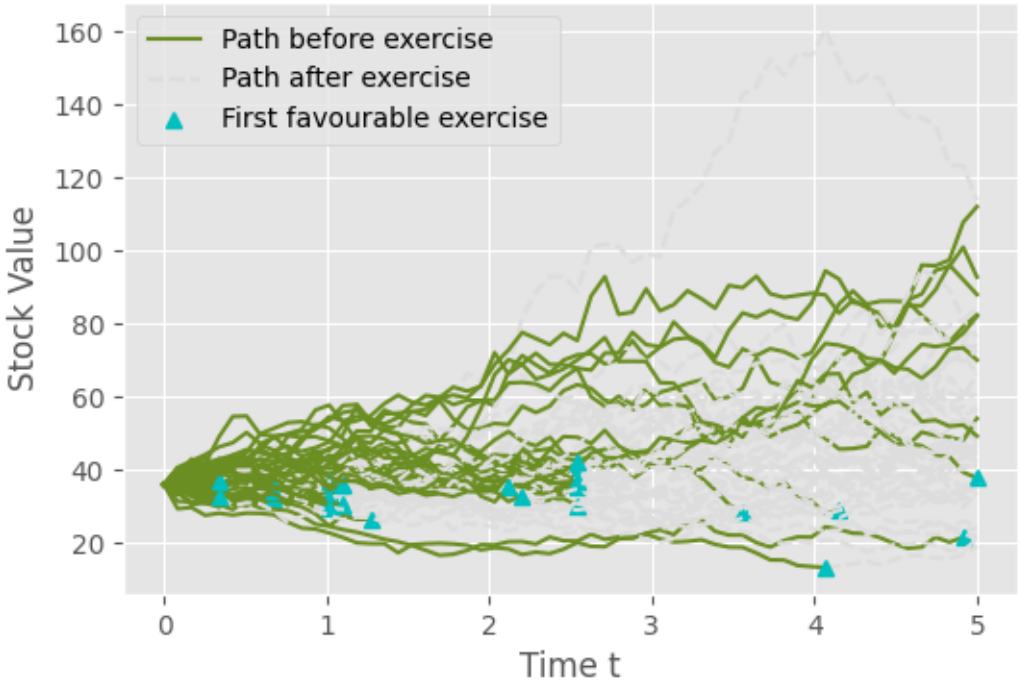

FIGURE 3.4: The optimal stopping decisions by the Least Square Monte Carlo Method

To estimate the condition expectation in equation 3.4, we regres with the basis functions taking on the underlying asset for the option being the independent variable:

$$F(\omega; t_{K-1}) = \sum_{j=0}^{\infty} a_j L_j(X)$$

where a is the coefficients for the regression, L is the basis function, where the argument is the underlying asset X (Longstaff and Schwartz, 2001).

The LSM approach gives a lower bound for the true price of the option given optimal stopping choice:

Proposition 3.2.1. ***Lower Bound To True Value:*** *For any finite choice of M, K, and vector* $\theta \in \mathbb{R}^{M\times(K-1)}$ *representing the coefficients for the M basis functions at rach of the K-1 early exercise dates, let* $LSM(\omega; M, K)$ *denote the discounted cash flow resulting from*

the following the LSM rule of exercising when the immediate exercise value is positive and greater tahn or equal to $\hat{F}_M(\omega_l; t_k)$ as defined by θ. Then the following inequality holds almost surely,

$$V(X) \geq \lim_{N\to\infty} \frac{1}{N} \sum_{i=1}^{N} LSM(\omega_i; M, K)$$

(p. 124 (Longstaff and Schwartz, 2001))

3.2.2 Numerical results

By the above two algorithms for valuation, we choose to vary spot, volatility and maturity for pricing an American put option with K=40 and r=0.06. This table will serve as reference for the machine learning algorithm in chapter (!TODO chapter for machine learning). For the binomial tree we use 100 time-steps, which gives stable results (compare to figure 3.2) and for the LSM we use 10^5 paths with 50 time-steps per year. The European option is valued by using BS closed form solution for a call option (see proposition 3.2.1) and Put-call parity (see proposition **??**).

TABLE 3.1: Valuation of American put option with K=40 and r=0.06.

Spot	σ	**T**	**Closed form European**	**Binomial Tree**	**LSM**	**abs. diff.**
36	0.2	1	3.844	4.488	4.478	0.010
36	0.2	2	3.763	4.846	4.828	0.018
36	0.4	1	6.711	7.119	7.092	0.027
36	0.4	2	7.700	8.508	8.500	0.008
38	0.2	1	2.852	3.260	3.245	0.015
38	0.2	2	2.991	3.748	3.735	0.013
38	0.4	1	5.834	6.165	6.144	0.021
38	0.4	2	6.979	7.689	7.665	0.024
40	0.2	1	2.066	2.316	2.313	0.003
40	0.2	2	2.356	2.885	2.881	0.004
40	0.4	1	5.060	5.310	5.326	0.016
40	0.4	2	6.326	6.914	6.908	0.006
42	0.2	1	1.465	1.622	1.622	0.000
42	0.2	2	1.841	2.217	2.212	0.005
42	0.4	1	4.379	4.602	4.596	0.006
42	0.4	2	5.736	6.264	6.243	0.021
44	0.2	1	1.017	1.117	1.113	0.004
44	0.2	2	1.429	1.697	1.688	0.009
44	0.4	1	3.783	3.956	3.962	0.006
44	0.4	2	5.202	5.656	5.649	0.007

We see the maximum difference between the two algorithms is 0.027 at S=38, $\sigma = 0.4$ and T=2. The other obvious fact is that the European put has a lower value than its American counterpart, because the continuous exercise feature adds additional value to the put option.

3.3 Benchmarks in higher dimensions

In this section will we provide closed form solution for some special cases of european multivariate contingent claims. Furthermore we present a lattice approach in multidimensional for pricing both european and american multivariate contingent claims. The basic assumptions and results are given in section 2.2.

3.3.1 Analytical formulas for Rainbow options

We derive closed form solutions to european call and put options depending on several variables, for simplicity we will focus on pricing options with 2 or 3 underlying stocks. We apply the intuition given in (Johnson, 1987) and the results given in (Ouwehand, 2006). The derivatives we will consider are the geometric mean -, maximum - and minimum call option.

3.3.1.1 Geometric basket call option

For a geometric basket call option the contract function is given by:

$$\Phi(S(T)) = \max\{(\prod_{i=1}^{n} S_i(T))^{\frac{1}{n}} - K, 0\}$$

The key to derive a closed form solution is the known result that the sum of normal random variables are multivariate normal distributed. This implies that the product of lognormal random variables are multivariate log-normal distributed. Since:

$$\exp(x+y) = \exp(x) \cdot \exp(y)$$
$$X \sim \mathcal{N}(\mu, \sigma^2) \Rightarrow Y = \exp(X) \sim LN(\mu, \sigma^2)$$

We assume as in section 2.2 that the stocks price process follows a GBM, hence:

$$(\prod_{i=1}^{n} S_i(T))^{\frac{1}{n}} = (\prod_{i=1}^{n} S_i(0))^{\frac{1}{n}} \exp((r - \frac{1}{2n}\sum_{i=1}^{n}\sigma_i^2)T + \frac{1}{n}\sum_{i=1}^{n}\sigma_i W_i(T)) \tag{3.5}$$

By defining

$$\sigma = \frac{1}{n}\sqrt{\sum_{i=1}^{n}\sigma_i^2 + 2\sum_{i\neq j}\rho_{i,j}\sigma_i\sigma_j} \tag{3.6}$$

$$F = (\prod_{i=1}^{n} S_i(0))^{\frac{1}{n}} \exp((r - \frac{1}{2n}\sum_{i=1}^{n}\sigma_i^2)T + \frac{1}{2}\sigma^2 \cdot T \tag{3.7}$$

We arrive at the price by skipping some arguments:

$$\Pi(t, \mathcal{X}) = \exp(-r * (T-t))\Big(FN(d_1) - KN(d_2)\Big) \tag{3.8}$$

where $d_1 = \frac{\ln(\frac{F}{K}) + \frac{1}{2}\sigma^2 T}{\sigma \cdot \sqrt{T}}$ and $d_2 = d_1 - \sigma\sqrt{T}$

3.3.1.2 Options on the Maximum or the Minimum of Several Assets

Here we restrict ourselves to consider the case with three underlying stocks like in (Boyle, Evnine, and Gibbs, 1989) and (Ouwehand, 2006), but the formula can be generalized to higher dimensions. The contract functions we consider are:

- Best of assets or cash: $\Phi(S(T)) = \max\{S_1, S_2, \ldots, S_n, K\}$
- Call on max: $\Phi(S(T)) = \max\{\max(S_1, S_2, \ldots, S_n) - K, 0\}$

We use n=3 because it shows the generality without the notation becomes to cumbersome.

We use the martingale framework developed in section 2.2 to value these exotic options. The key is to choose the numeraire to a risky assets instead of the bank account. By results from section 2.2 the processes are still Q-martingales given the numeraire is strictly postive. So under the asssumption the arbitrage free and complete market it follows:

$$S_0(t)E_t^{Q_0}[\frac{X_T}{S_0(T)}] = S_1(t)E_t^{Q_1}[\frac{X_T}{S_1(T)}]$$

3.3.1.2.1 Best of assets or cash

The best of assets will both provide a price and the method for pricing call on max and min. We assume WLOG n=4 and define the payoff as for the i'th asset:

$$S_i(T) \cdot 1_{\{S_i(T)>S_j(T):i\neq j\}}$$

Hence the best of assets derivative is a sum of above equation for each asset. So we are considering four cases, because we assumed WLOG n=4.

For i=1 we set S_1 to be the numeraire asset with martingale measure $\mathbb{Q}_1$. Then we see by using RNVF (see proposition 2.2.4):

$$\begin{aligned}\Pi_1(t,\mathcal{X}) &= S_1(t)E_t^{Q_1}[1_{\{S_1(T)>S_2(T),S_1(T)>S_3(T),S_1(T)>S_4(T)\}}] \\ &= S_1(t)Q_1[\ln(\frac{S_2(T)}{S_1(T)}) < 0, \ln(\frac{S_3(T)}{S_1(T)}) < 0, \ln(\frac{S_4(T)}{S_1(T)}) < 0]\end{aligned} \tag{3.9}$$

By cycling through the numeraires we get four derivatives that we need to add together for optaining the fair price for best of assets $\Pi_{max}(t,\mathcal{X})$. Before we can proceed we need to find the probaility under the Q-martingale measure. By using Ito's lemma (see A.0.1):

$$\ln(\frac{S_i(T)}{S_j(T)}) \sim \mathcal{N}(\ln(\frac{S_i(T)}{S_j(T)}) - \frac{1}{2}\sigma_{i/j}^2 \cdot (T-t), \sigma_{i/j}\sqrt{T-t})$$

where $\sigma_{i/j}^2 = \sigma_i^2 + \sigma_j^2 - 2\rho_{ij}\sigma_i\sigma_j$.

Besides using the definition for d_1 and d_2 in proposition 3.2.1 we define:

$$d_1^{i/j} = \frac{1}{\sigma \cdot \sqrt{T-t}} \cdot \left(\ln(\frac{S_i}{S_j}) + \frac{1}{2}\sigma_{i/j}^2 \cdot (T-t)\right) \tag{3.10}$$

$$d_2^{i/j} = d_1^{i/j} - \sigma_{i/j}\sqrt{T-t} \tag{3.11}$$

The correlation between $\frac{S_i(T)}{S_k(T)}$ and $\frac{S_j(T)}{S_k(T)}$ is given by (see page 5 (Ouwehand, 2006)):

$$\rho_{ij,k} = \frac{\rho_{ij}\sigma_i\sigma_j - \rho_{ik}\sigma_i\sigma_k - \rho_{kj}\sigma_k\sigma_j + \sigma_k^2}{\sqrt{(\sigma_i^2 + \sigma_k^2 - 2\rho_{ik}\sigma_i\sigma_k)\cdot(\sigma_j^2 + \sigma_k^2 - 2\rho_{jk}\sigma_j\sigma_k)}} \tag{3.12}$$

Hence:

$$Q_1[\ln(\frac{S_2(T)}{S_1(T)}) < 0, \ln(\frac{S_3(T)}{S_1(T)}) < 0, \ln(\frac{S_4(T)}{S_1(T)}) < 0] = N_3(-d_2^{2/1}, -d_2^{3/1}, -d_2^{4/1}, \rho_{23,1}, \rho_{24,1}, \rho_{34,1})$$

Cycling through each derivative, we get:

$$\begin{aligned}\Pi_{max}(t,\mathcal{X}) = {} & S_1(t)N_3(-d_2^{2/1}, -d_2^{3/1}, -d_2^{4/1}, \rho_{23,1}, \rho_{24,1}, \rho_{34,1}) \\ & + S_2(t)N_3(-d_2^{1/2}, -d_2^{3/2}, -d_2^{4/2}, \rho_{13,2}, \rho_{14,2}, \rho_{34,2}) \\ & + S_3(t)N_3(-d_2^{1/3}, -d_2^{2/3}, -d_2^{4/3}, \rho_{12,3}, \rho_{14,3}, \rho_{24,3}) \\ & + S_4(t)N_3(-d_2^{1/4}, -d_2^{2/4}, -d_2^{3/5}, \rho_{12,4}, \rho_{13,4}, \rho_{23,4})\end{aligned} \tag{3.13}$$

We can extend the above result to best of assets and cash by letting $S_4(t) = K\exp(-r(T-t))$, where K do not have any volatilty and also independent of the other assets, hence (3.13) becomes:

$$\begin{aligned}\Pi_{max}(t,\mathcal{X}) = {} & S_1(t)N_3(-d_2^{2/1}, -d_2^{3/1}, d_1^1, \rho_{23,1}, \rho_{24,1}, \rho_{34,1}) \\ & + S_2(t)N_3(-d_2^{1/2}, -d_2^{3/2}, d_1^2, \rho_{13,2}, \rho_{14,2}, \rho_{34,2}) \\ & + S_3(t)N_3(-d_2^{1/3}, -d_2^{2/3}, d_1^3, \rho_{12,3}, \rho_{14,3}, \rho_{24,3}) \\ & + K\cdot\exp(-r(T-t))N_3(-d_2^1, -d_2^2, -d_2^3, \rho_{12}, \rho_{13}, \rho_{23})\end{aligned} \tag{3.14}$$

3.3.1.2.2 Call on max and call on min

From (3.14) is easy to see the call max fair price is:

$$\begin{aligned}\Pi_{cmax}(t,\mathcal{X}) = {} & S_1(t)N_3(-d_2^{2/1}, -d_2^{3/1}, d_1^1, \rho_{23,1}, \rho_{24,1}, \rho_{34,1}) \\ & + S_2(t)N_3(-d_2^{1/2}, -d_2^{3/2}, d_1^2, \rho_{13,2}, \rho_{14,2}, \rho_{34,2}) \\ & + S_3(t)N_3(-d_2^{1/3}, -d_2^{2/3}, d_1^3, \rho_{12,3}, \rho_{14,3}, \rho_{24,3}) \\ & - K\exp(-r(T-t))\cdot\left(1 - N_3(-d_2^1, -d_2^2, -d_2^3, \rho_{12}, \rho_{13}, \rho_{23})\right)\end{aligned} \tag{3.15}$$

To derive put max we can utilize a put-call-parity (see page 6 (Ouwehand, 2006)), but it takes a different form than the one presented in 2 (see 2.3.2). The relationship for the exotic call options:

$$V_c(K) + K\exp(-r\cdot(T-t)) = V_p(K) + V_c(0)$$

Where $V_c(K)$ is the value of the exotic call option.

These options will serve as benchmark for the multivariate lattice approach, and for pricing the call on min see (Ouwehand, 2006).

3.3.2 Lattice approach for multivariate contingent claims

We follow the approach in (Boyle, Evnine, and Gibbs, 1989) (BEG method), because it is the natural extension of the Cox Ross Rubinstein model (section 3.1) for multivariate contingent claims. The idea as in the one dimensionel case is to approximate the system of underlying processes (assumed to be GBMs) with a discrete multivariate binomial lattice. The advantage is that for exotic options like the rainbow options the valuation of european put options is readily extented to the american put options and has high accuracy. The (Boyle, Evnine, and Gibbs, 1989) has its limitation in terms of number of underlyings and for path dependent options (see section 3.1), but it is very intuitive and extends easily to american options. The problem with increasing the number of underlyings is that the number of one-step transition at each node is 2^d and the total number of terminal nodes after N steps is $(N+1)^d$ for path-independent derivatives, which means for high dimensional problems the computational resources become an issue with this discrete approximation approach. This makes the BEG method undesirable for higher dimensions than three so we will focus on the two dimensional case. Another problem with two or more underlyings are that some one-step transition probabilities negative, which makes the model useless in those cases.

The model we want to approximate is the bivariate lognormal distribution, because we assume the Black Scholes model to describe the evolution of the two risky assets (section 2.2). We restrict ourselfes to the assumptions 2.3.1 given in the classical (Black and Scholes, 1973), hence for risk neutral pricing the SDE for the risky assets are:

$$dS_i(t) = S_i(t)r(t)dt + S_i(t)\sigma_i(t)dW^{\mathbb{Q}}(t) \quad for\ i = 1, 2, \ldots, d$$

We divide the time from inception to maturtiy (length T) into N equidistant intervals with length *Deltat*, because we want the jump distribution to approximate the continuous time multivariate lognormal distribution. Each time interval has a jump size defined in terms of the volatility and the length of the interval:

$$u_i = \exp(\sigma_i\sqrt{\Delta t}) \quad and \quad u_i \cdot d_i = 1 \quad for\ i = 1, 2, \ldots, d$$

The u_i and d_i are the multiplication factor for the i'th stock, where the former is a jump up and the latter is a jump down for the stock. With what probability does the stock jump up or down? This is the key in the BEG approach to approximate a multivariate lognormal distribution with a discrete distribution. The probabilities are chosen such that the characteristics functions are equal for small time steps *Deltat* (see p. 245-246 in (Boyle, Evnine, and Gibbs, 1989) for details). The probabilities for the model with two underlying risky assets:

$$\begin{aligned}
p_1 = p_{uu} &= \frac{1}{4}\left(1 + \rho + \sqrt{\Delta t}\left(\frac{\mu_1}{\sigma_1} + \frac{\mu_2}{\sigma_2}\right)\right) \\
p_2 = p_{ud} &= \frac{1}{4}\left(1 - \rho + \sqrt{\Delta t}\left(\frac{\mu_1}{\sigma_1} - \frac{\mu_2}{\sigma_2}\right)\right) \\
p_3 = p_{du} &= \frac{1}{4}\left(1 - \rho + \sqrt{\Delta t}\left(-\frac{\mu_1}{\sigma_1} + \frac{\mu_2}{\sigma_2}\right)\right) \\
p_4 = p_{dd} &= \frac{1}{4}\left(1 + \rho + \sqrt{\Delta t}\left(-\frac{\mu_1}{\sigma_1} - \frac{\mu_2}{\sigma_2}\right)\right)
\end{aligned} \tag{3.16}$$

The correlation ρ between the two assets are assumed to be constant and $\mu_i = r - \frac{1}{2}\sigma_i^2$. We have illustrated below a two-dimensional lattice, where we see that the number of nodes at maturity is $(1+N)^d$.

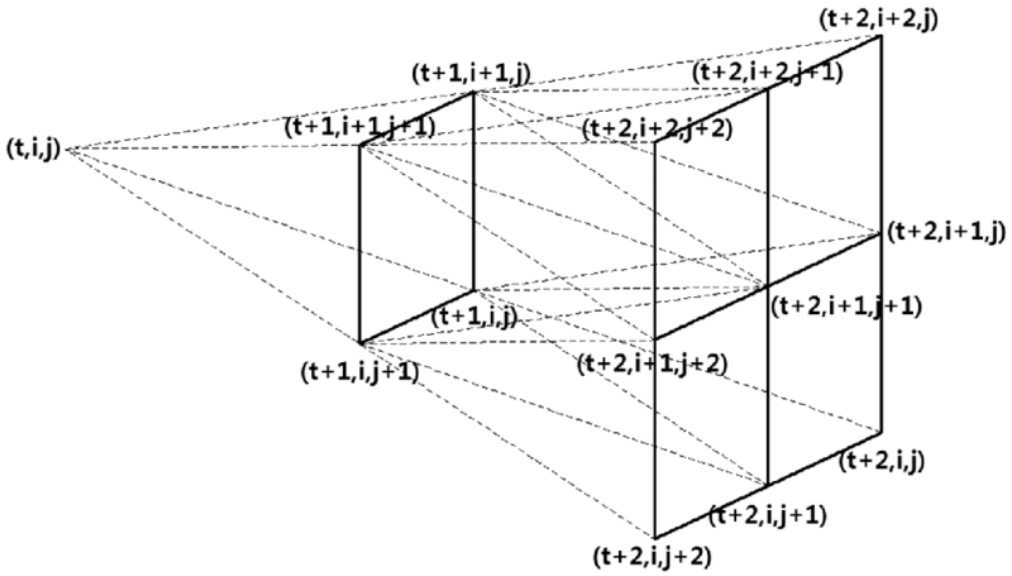

FIGURE 3.5: Evolution of binomial model with two underlying risky asset, where t is time, i is number of up movement for S_1 and j is number of up movement for S_2 (**Kim02**).

After the construction of the evolution of the underlyings, we can like in the one dimensionel CRR model recursively working backward in the multidimensional binomial lattice. For the european option the recursive formula is:

$$V_{i_1,\ldots,i_d}(t) = \exp(-r\Delta t)\Big(p_1 V_{i_1+1,\ldots,i_d+1}(t+1) + \cdots + p_{2^d} V_{i_1,\ldots,i_d}(t+1)\Big)$$

For the american option we have the possibility of exercising between inception and maturity of the contract hence:

$$V_{i_1,\ldots,i_d}(t) = \max\{\Phi(t,\boldsymbol{S}(t)), \exp(-r\Delta t)\Big(p_1 V_{i_1+1,\ldots,i_d+1}(t+1) + \cdots + p_{2^d} V_{i_1,\ldots,i_d}(t+1)\Big)\}$$

With the recursive formulas we can valuate multivaraite contigent claims for a variety of exotics including the american put option with several underlying risky assets. The increasing dimensions also increase the number of one-step transition probabilities, hence the (Ekvall, 1996) approach (NEK) tries with setting all the probabilities equal to 0.5 and than determine the jump sizes. The NEK approach overcome the issue with negative probabilities for higher dimensions and following (Ekvall, 1996) the algorithm seems also to have faster convergence. The NEK and BEG approaches are both good for low dimensionel option problems, but both still suffers the curse of dimensionality. The simulations methods on the otherhand has somewhat an advantage for higher dimesions, but e.g. the LSM needed basis function grows exponential with the underlyings. We choose to present the BEG approach, because it is a natural extension of the already presented CRR model, which we already build

inituition opon.

3.3.3 Numerical And Analytical Results

In table 3.2 we look how the binomial lattice approach to multivariate contingent claims performs compared with the analytical solution also derived in this section. For the exotic european options we have a closed form solution, so we can measure how the binomial lattice approach works in those cases. With solutions from the lattice approach within a small range of the closed form solution gives us confidence in the multivariate binomial model for pricing e.g. american put options, where there is no closed form solution.

TABLE 3.2: Valuation of multivariate contigent claims with two underlyings with K=40, $S_1(0) = S_2(0) = 40$, $\sigma_1 = 0.2, \sigma_2 = 0.3$, T=1, $\rho = 0.5$ and r=0.06.

Derivative type	**Method**	**No. Steps**	**Price**
European Put Minimum	BEG	10	4.248
	BEG	50	4.341
	BEG	100	4.352
	Analytic form		4.363
European Call Minimum	BEG	100	2.475
	Analytic form		2.483
European Call Maximum	BEG	100	7.787
	Analytic form		7.800
American Put Minimum	BEG	10	3.830
		50	3.884
		100	3.890

From table 3.2 we see that the BEG approach has highest accuracy for 100 equidistant time-steps hence we choose to price with N=100 for the multivariate binomial model. The biggest absolute difference is 0.013 which is reasonable within the market spread for trading. The results are promising for valuation of american put options based on several underlying assets. This table will serve as reference and benchmark for the more recent approaches with neural networks in chapter 6.

Chapter 4

Deep Learning

The chapter is inspired by (Goodfellow, Bengio, and Courville, 2016) which the interested reader can find more information about deep learning. Deep learning experiences a renaissance, because of the technology improvements in hardware and software. The collection of data has also significant improved the field. Deep learning is a specialized field in machine learning, where you focus on a special architecture of models. Like in machine learning the basic components of a deep learning algorithm are a dataset, cost function, optimazation algorithm and a model. E.g. in the lsm method we assumed the model was gaussian, dataset was the simulated paths, the cost function was the mean square error and the optimazation algorithm was a closed form solution of the normal equations. Deep learning is about studying neural networks which allows for greater flexiblity than standard methods like linear regression. "Deep" comes from that a neural network consists of multiple layers, where the depth tells you how many layers the network has (see figure 4.2). All the algorithms applied will be within supervised learning, where we try to fit the best relationship between the features and the response variable. Furthermore all our algorithms will be based on the multilayer perceptrons (MLPs) for regression, hence it will also be the main focus. The advantage of a multi-layer model is that for each layer the updated set of covariates can be more finely tuned to better explain the data. These network of models are called feedforward because the information only travels forward in the neural network, through the input nodes then through the hidden layers (single or many layers) and finally through the output nodes. First we present the basics for machine learning and then specialize the theory to deep learning.

4.1 Machine Learning Basics

The task for machine learning is to learn from data with respect to some performance measure. "A computter program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E" ((Goodfellow, Bengio, and Courville, 2016) p. 97). Classical tasks T could be classification or regression, where the two methods differ on the output type. The former has discrete outputs, where regression has continuous output. We are interested finding a price, which is naturally represented as a continuous value hence our task will be to regress the price or continuation value like in LSM. Measuring performance depends on the task, but for regression a typical performance measure is mean squared error (MSE):

$$\frac{1}{n}\sum_{i=1}(y_i - \hat{y}_i)^2$$

There are multiple methods how to penalize the error e.g. mean absolute error (MAE). Another measure to quantify the fit is coefficient of determination:

$$R^2 = 1 - \frac{\frac{1}{n}\sum_{i=1}(y_i - \hat{y}_i)^2}{\frac{1}{n}\sum_{i=1}(y_i - \bar{y})^2}$$

Coefficient of determination explains how well the model explains the data compared to estimating by the empirical mean. Some care should be taken by comparing models with different capacity, because the coefficient of determination will tend to prefer the larger models. The experience comes from data, where the data can be given with or without target values $\boldsymbol{y}$. In machine learning the task is quite different depending on targets are given or not, hence the algorithms are split into two categories supervised and unsupervised learning. The terminology supervised comes from a teacher gives you the target values $\boldsymbol{y}$ that the algorithms tries to predict from $\mathbf{X}$. The models for our purposes will be supervised regression models.

Machine learning is not about making overly complex models to fit your data perfectly, because then the model will most likely fit new data badly. This phenomenon is known as overfitting, but the models can also be too simpel known as underfitting. If machine leearning only cared about performance on the given data for training, then machine learning would be essentially optimization. The key difference is that machine learning wishes to obtain statistical generalization to unseen data. The practical way to measure generalization error is to measure it on a test set. In machine learning the data is split into test data and training data. The training data is used for training the model, where the training error tells how the fitted parameters fits to the training data. A unacceptabel high training error could indicate the regresson method could be too simple and underfitting is the issue. The test data is used for measuring the model performance on unseen data, which is essentially what we want to be within an acceptable range. An unacceptable test error and low training error could be an example of overfitting the model. Hence when training the model, we best of both worlds acceptable the training error and making the gap between training and test error acceptable as well.

A techinique known as regularization is one approach to reduce test error. The regularizer is added to the cost function $J(\theta)$, which is essentially the function to minimize in order to train the model. There is a lot different regularization methods, where in the MLPs section 4.2 we will present some useful regularization for deep learning models. For training it can sometimes also be useful to have a validation data set to see how your model generalizes, because the test set cannot be used to make model choices. Besides the parameter estimated within the model, there is a need for model designs (hyperparameters). I.e hyperparameters is determine outside the learning algorithm where one example could be model capacity.

Hyper

The parameters within the model are found by minimizing the cost function $J(\theta)$. In some cases there exist either a closed form solution or the cost function is convex making the optimization problem easier to handle. The complexity of MLPs makes the cost function nonconvex and iterative optimization procedures are needed. The basic iterative procedure is the gradient descent where the concept is to reapetedly making small moves in parameters toward better configuration. The most popular gradient methods in deep learning are Adam, RMSProp, AdaGrad and stochastic

gradient descent (SGD). The methods will be discussed in more details in section 4.2.3. To sum up a machine learning model needs a dataset, a model, a cost function and an optimization algorithm. Understanding of basic machine learning will be the foundation for deep learning.

4.2 Multilayer Perceptrons

The goal of the multilayer perceptrons (MLPs) is to approximate a function $f^*(x)$, where the MLPs defines a mapping $f(x;\theta)$ to approximate $f^*(x)$. The task is to find the best θ such that the approximatation $f(x;\theta)$ is close to the targets measured by a defined cost function $J(\theta)$. With the minimized cost function the goal is to archive statistical generalazation i.e. useful results for test data not used for training.

The first step for MLPs is building the network, where we start with zooming in on a single neuron, which is one node of a directed acyclic graph (see figure 4.2). Note the MLPs is called a feedforward network, because all the connections between the neurons are directed such that the network forms a directed acyclic graph.

4.2.1 A Single Neuron

The single neuron has a number P features of inputs x_p and one output $\hat{y}$ (see figure 4.1).

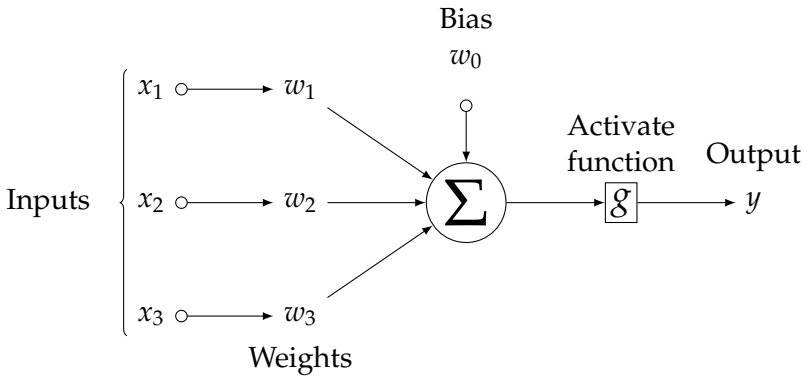

FIGURE 4.1: A single neuron

The function from inputs to output for a single neuron is:

$$\hat{y} = g(w_0 + \boldsymbol{x}^T\boldsymbol{w}) \quad where \quad \boldsymbol{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_p \end{pmatrix} \quad and \quad \boldsymbol{w} = \begin{pmatrix} w_1 \\ \vdots \\ w_p \end{pmatrix} \tag{4.1}$$

The $\boldsymbol{w}$ is the weight matrix (this case a vector) and w_0 is the bias term. The term inside the function g is the activation of the neuron and it is a affine transformation denoted:

$$a = w_0 + \boldsymbol{x}^T\boldsymbol{w}$$

The function g is the activation function and it is essential for the flexibility of the MLPs (MacKay, 2018). There exists numerous of activations function only the imagination is the limit. We will list the most common and discuss them.

4.2.1.1 Activation functions

Activation functions are important for neural network, because they allow for non-linearities and flexibility. Activation functions apply a non-linear transformation and decide whether a neuron should be activated or not. Without activation functions or the identity function $g(a) = a$ the whole network would essentially be a linear regression model. Some popular activation functions:

- Sigmoid function: $g(a) = \frac{1}{1+\exp(-a)}$

  This is the traditional choice, also called logistic function. Popular in classification but can suffer from vanishing gradient in deep learning.

- Hyperbolic tangent function: $g(a) = \frac{e^a - e^{-a}}{e^a + e^{-a}}$

  A scaled and shifted sigmoid function and likewise suffers from the vanishing gradient problem. The range is $(-1, 1)$ and centered at zero. Often used for hidden layers.

- ReLU function: $g(a) = \max(0, a)$.

  Rectified Linear Unit is one of the most popular choices since it does not suffer from the vanishing gradient problem. The MLPs often becomes more sparse because it sets some features to zero. It is claimed that ReLU learns muliple times faster than both the hyberbolic tangent and the sigmoid function.

- Leaky ReLU function:

$$g(a) = \begin{cases} a & if\ a \geq 0 \\ \alpha \cdot x & otherwise \end{cases}$$

  The fact that ReLU can have some hidden covariates that is zero which can be an advantage. A disadvantage of ReLU is some neurons may die out if the neurons are mapped to zero. The Leaky ReLU is designed to give such neurons a chance to get back into action, but not too easily, so $\alpha > 0$ is chosen small (typical $\alpha = 0.01$)

- ELU - exponential linear unit:

$$g(a) = \begin{cases} a & if\ a \geq 0 \\ \alpha(\exp(a) - 1) \cdot x & otherwise \end{cases}$$

  Like the Leaky ReLU the ELU is designed to avoid the dead ReLU problem (Goodfellow, Bengio, and Courville, 2016).

With a understanding of a single neuron we continue to the architecture of a MLPs.

4.2.2 Architecture Of MLPs

A MLPs consists of a input layer, where all p features enter, L hidden layers, and an output layer. Each hidden layer and the output layer consists of multiple neurons, where the width of the layer is the number of neurons in that layer (m^l) (see figure 4.2). The networks inputs are called the input layer, the output layer is the output of the neural network. The layers between the input and output layer are hidden layers. This could be an explanation why the field is called Deep learning, because of a deep structure of layers. In each hidden layer a linear combination of the features from the previous layer is made and then an activation function is applied in order to create the new hidden features in that layer (see figure 4.2). The output in MLPs is a larged nested function, where the input layer go through a chain of functions until reaching the output.

$$f(\boldsymbol{x};\boldsymbol{\theta}) = f_1 \circ f_2 \circ \cdots \circ f_{L+1} \tag{4.2}$$

$$\textit{where } f_i : \mathbf{R}^{m^{i-1}} \to \mathbf{R}^{m^i} \quad i = 1, \ldots, L+1 \tag{4.3}$$

Each function in the composition of functions corresponds to a layer of neurons.

$$f_i(x) = g(\mathbf{W}^T \boldsymbol{x} + \boldsymbol{w_0}) \quad x \in \mathbf{R}^{m^{i-1}} \textit{ and } \mathbf{W} \in \mathbf{R}^{\boldsymbol{m^{i-1}} \times \boldsymbol{m^i}} \tag{4.4}$$

So the function maps a vector to a vector, the hidden layers will often be denoted $\boldsymbol{h}$ where for each single neuron, we map a vector to a scalar by:

$$h_i = g(\boldsymbol{x}^T \cdot \mathbf{W}_{:,i} + (w_0)_i)$$

A layer is a vector of neurons, hence the transformation from one layer to the next can be interpreted as multiple vector to skalar transformations, where each skalar/neuron acts in parallelle. The different view motivates the initial presentation of single neuron network (section 4.2.1).

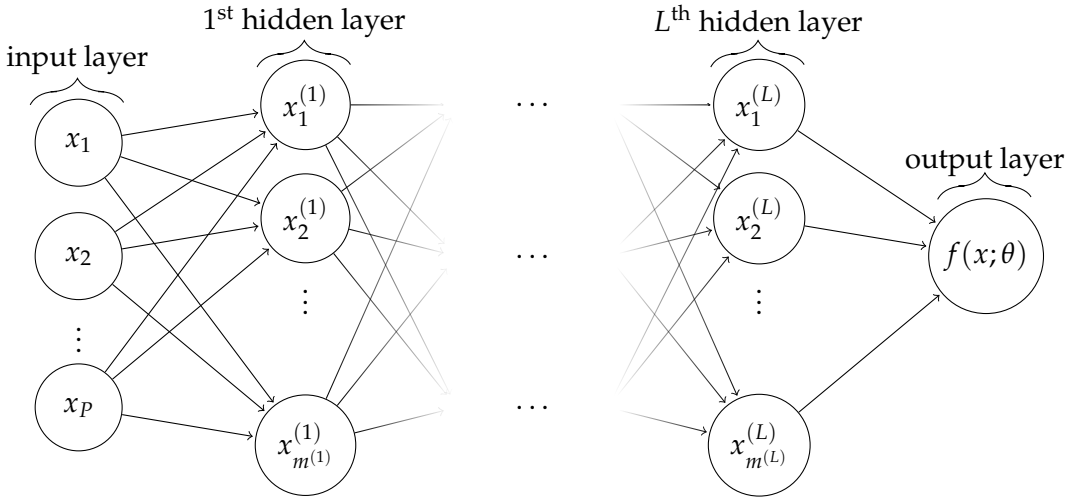

FIGURE 4.2: Multilayer perceptrons with $(L+1)$-layers with P input features and 1 output. The l^{th} hidden layer contains $m^{(l)}$ hidden neurons.

So the MLPs is not like classical linear regression in section 3.2 where a single linear transformation from input to output is applied. The unique attribute of neural network is the ability to approximate any kind of function (Universal Apporximate

Theorem page 194 (Goodfellow, Bengio, and Courville, 2016)), because of the flexiblility with applying multiple functions to the input layer. The neural network has a lot of different design options, where e.g. hidden layers, layer width, deepth, activation functions etc. are hyperparameters. (TODO) In general it is recommended to use too many hidden covariates (neurons) rather than too few, and instead introduce some kind of penalty to avoid that the model becomes overly large, which we will discuss further in section 4.2.4.

To fit a model the model need initialization of weights, biases and activation functions. In order to measure the performance of the model, we need a function to measure the difference between the approximation $f(\boldsymbol{x};\boldsymbol{\theta})$ and the target value $f^*(\boldsymbol{x})$. This function is referred as the loss function, where the cost function is the average over the loss functions. The cost function tells how close the predition $f(\boldsymbol{x};\boldsymbol{\theta})$ is to the target y. The cost function is key to improving our model or in machine learning lingo training the model, hence the next section will cover model training (Goodfellow, Bengio, and Courville, 2016).

4.2.3 Training The Network

Training the network is key to obtaining for building a useful model. The performance of the model is measured by the cost function. One example of a cost function is the risk function:

$$J(\theta) = E_{(\boldsymbol{x},y)\sim p_{data}} L(f(\boldsymbol{x};\boldsymbol{\theta}), y)$$

The true probability function generating the data (p_{data}) is unknown to us, hence the empirical risk function is often chosen:

$$J(\theta) = E_{(\boldsymbol{x},y)\sim \hat{p}_{data}} L(f(\boldsymbol{x};\boldsymbol{\theta}), y) = \frac{1}{n}\sum_{i=1}^{n} L(f(x_i;\theta), y_i)$$

I.e. the cost function is a way to measure the approximation of $f^*(x)$ by our model $f(\boldsymbol{x};\boldsymbol{\theta})$. The training is important e.g. imagine after random initialization of parameter and construction of the MLPs we reported the output given the inputs of the model. This model would probably results in a high cost function value and bad performance because the given weights would produce a function that do not fit training data. The way out of the high cost function is to try to minimize the cost function over the weightspace hence training for MLPs is essentially a optimization of a nonconvex function. Remember the MLPs is a chain of functions (section 4.2.2) where the structure often makes the optimization a nonconvex problem hence a global minimum is seldom archived. Other pitfalls for optimization of MLPs are weight space symmentry (nonidentifiable), steep cliffs, saddle points, vanishing and exploding gradient.

The actual optimization algorithms for MLPs are based on gradients, where we make small local moves. The overall goal is to find θ to reduce the test error. Within gradient methods there are batch gradient descend and minibatch stochastic gradient descend, where the former is training on the whole dataset, and the latter is only for a subset of the dataset. The minibatch methods have the advantage it can parrallize hence faster training. An epoch is the number of complete dataset training cycles to update the weights. For the minibatching techniques it is important to

random sample from the whole dataset in order to get unbiased gradient estimation.

Common method to estimate the parameters is gradient descent, stochastic gradient descent (SGD) and Adam, where all the optimization algorithms are iterative. The goal is to find critical points $\nabla J(\theta) = 0$, to obtain the critical points the iterative methods move in the opposite direction of sign of derivative $\nabla J(\theta)$. I.e. the move updates the parameters where the stepsize is our learning rate η:

$$\theta_{new} = \theta_{old} - \eta \nabla J(\theta_{old})$$

$J(\theta)$ is the cost function, where the choice for the models we consider will be the empirical risk measure, where the loss function L will be quadratic:

$$J(\theta) = \frac{1}{n} \sum_{i=1}^{n} L(f(x_i; \theta), y_i) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

Gradient descend uses the whole batch for each update, where Adam and SGD use minibatches for each update. The Adam algortihm use a adaptive learning rate, where the two others use constant or descending learning rate. The Adam method makes greater progress in more gently sloped directions of weightspace compared to SGD and gradient descent.

Besides choosing a optimization procedure the initialization of the parameters are important. There is a lot different suggestions to initialize parameters, but there is no general golden rule at the moment because lack of understanding of the optimization procedure in neural nets. Often the practioners tend to use simple and heuristic methods where it has been shown the initialization needs to break symmetry.

The most common way of finding gradients is the backpropagation algoritm, where the basic idea is the chain rule from calculus: $\frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx}$
To understand backpropagation is often useful with a computational graph created by forward propagation, where the backpropagation compute the derivative from output layer to input layer by going backward in the computational graph. The training process is a forward-backward algorithm. Different starting values of θ will result in different parameters. The good news is that these predictors typically do not differ by very much. It is recommended to work with a set of different starting values, and then use as a final predictor the average of the individual predictors stemming from each starting value (Goodfellow, Bengio, and Courville, 2016).

4.2.4 Regularization

The number of parameters and the capacity of neural networks arise often the problem of overfitting, i.e. the model does not generalize well on new data. Regularization is a technique that constains our optimization problem to discourage complex models, hence avoid the problem of overfitting. Some common methods for deep learning are parameter norm regularization, early stopping and dropout.

Early stopping is a effective and simple method for regularization. Compared to parameter norm regularization the early stopping algorithm does not harm the learning dynamics. The early stopping method is also computational efficient, hence

it makes a popular regularization method for deep learning. The idea of early stopping is that the iterative training algorihm keeps improving the train error, but training too extensively leads the test error to rise. The idea is then to split your data into validation and train datasets, where you determine the best $\hat{\theta}$ and the corresponding training steps $\hat{i}$ by iterative comparing the cost function on the validation set. The algorithm stops after a predefined number of steps without improving the cost function on the validation set.

Like early stopping the dropout method is computationally inexpensive. The idea is to remove neuron randomly at each training loop, i.e. when updating the gradient, each node is kept with probability p, independently of each other. To perform dropout a binary mask is sampled independently for each iterations, where the probability p for 1 is another hyperparameter. The goal is to minimize $E_{\mu}J(\boldsymbol{\theta}, \boldsymbol{\mu})$ where $\boldsymbol{\mu}$ is the mask. The result of the procedure is more robust features and a regularizing effect on most models.

There is a lot of design options for deep learning, where the choices should be specific for the given task. There is a no free luncg theorem for machine learning, which says no model is superior for all tasks (see page 114 (Goodfellow, Bengio, and Courville, 2016)). The design for specific tasks are important where training error and test error can be improved by designing the model for the given task. Another aspekt is the computational resources i.e. the memory space and computational time.

Pricing derivatives with deep learning methods have two clear benefits. The first is computational time, where after a model is trained, then the model is far superior to methods as monte carlo simulation or similar. Another advantage is the non-linearities of the model making it possible to fit more complex functions (Goodfellow, Bengio, and Courville, 2016).

Chapter 5

Option Pricing And Deep Learning

Deep learning can be applied to option valuation in different ways. After the two sections with MLPs calibrated to pricing models the MLPs will be applied to approximate optimal exercise strategy similar to LSM. The second method is to simulate input parameter, i.e. the market parameters and model parameters and then using a classical method for calculating target values $\boldsymbol{y}$ for the given input parameters $\mathbf{X}$. The method falls within supervised regression where we will use a MLPs network introduced in section 4.2 to approximate the mapping. The risky assets are modelled with black scholes theory from earlier chapters hence the generations of labels for european and american stock options are already presented (Chapter 2, 3). The theory in chapter 4 will be specialized for the specific task and discussed. The supervised MLPs regression will be used to valuate european call options and american put options. The advantage of MLPs is the model can easily be extendend to high dimensional data, where the e.g. polynomial regression (section 3.2) is prone to overfit and slow compared to MLPs.

5.1 Multilayer Perceptrons Regression For Optimal Stopping

The first application of neural network is to investigate, if the LSM method can be improved by using MLPs regression to approximate continuation value instead of using the linear model in LSM. This section is influenced by (Longstaff and Schwartz, 2001; Kohler, Krzyżak, and Todorovic, 2010), where the approximate scheme with only in-the-money paths (ITM) is inspired by (Longstaff and Schwartz, 2001) and the idea of using neural network instead of the linear model comes from (Kohler, Krzyżak, and Todorovic, 2010). The algorithm is based on approximating the american put option with one or several underlyings by considering a bermuda option, which converge the the american option by making the equidistant time-steps sufficiently small.

The continuous time problem for american options is given as a optimal stopping problem:

$$\sup_{\tau \in \mathcal{T}([0,T])} E^{Q}[\exp(-r\tau)g(x_\tau)] \tag{5.1}$$

Where τ is a stopping time (definition 2.4), $\mathcal{T}([0,T])$ is the class og all [0,T]-valued stopping times, g(x) is the intrinsic value of the option when the underlying state is x and the risk neutral state process assumed to be markov $\{x_t \in \mathbb{R}^d | 0 \leq t \leq T\}$. The continuous problem is often approximated by a discrete version, where the

computational procedures can be applied, hence the american option is approximated by a bermuda option. For simplicity we assume that the state process is time-homogeneous and we model a discrete pricing problem. The reformulation of the optimal stopping problem in discrete time is:

$$\sup_{\tau \in \mathcal{T}(0,\dots,T)} E^{Q}[\exp(-r\tau)g(x_\tau)]$$

We introduce the value function $J_n(x)$ which is the value of the option at time n if the underlying state process is equal to x Universal approximate theorem

5.2 Multilayer Perceptrons Regression Pricing From Existing Methods

The MLPs pricing method has a different approach than the other methods, because it is only data drivin in the sense it is model free. The MLPs could easily be used to real data, which is investigated in (Gaspar, Lopes, and Sequeira, 2020). We revisit the work from (Hirsa, Karatas, and Oskoui, 2019), where we try to extend the pricing models to options with two underlying risky assets. The model will be the classical GBM model presented in earlier chapters. By choosing the GBM model the MLPs pricing method is ready for investigation both for vanilla options and exotic options. We stress that the MLPs pricing method is not restricted to GBM assumption, but can be applied to other models such as Hesten, Variance Gamma and real market data. The advantages with MLPs are the fast calibration and parameter estimation compared to classical methods binomial lattice and LSM. With the increased speed for pricing it can cost accurary specially if the data is sparse, which can arise when using the method for exotic options on real market data. For practical application the accuracy is severe if the predicted price is not within the bid-ask spread, because the aim is to price fast without introducing arbitrage. We will present results for european, american and other exotic options presented in earliar chapters with MLPs pricing methods.

For deep learning the hyperparameters is important for finding the right model for pricing, where different choices will be empirical presented under training. For the given task polynomial regression can also be used, but we will see later why MLPs regression is preferred. The section is split into three sections "Data", "Training" and "Performance", where the european call, american put, european put minimum and american put minimum option will be presented.

5.2.1 Data

The generation of labels are the computational expensive part of the MLPs method, because the method needs enough samples to approximate the function f^* well. The upside after generation of labels is that the method is computationally fast and easy to implement with basic knowledge of deep learning. The labels will be generated by existing methods presented in chapter 3, where the input parameters will be quasi sampled with halton sequences.

For both the european call and the american put the parameter in-sample will be the same, where we remember the 5 parameters for pricing an european call option (proposition 3.2.1). The european call and american put option is a first order

homogeneous function in $(S(0), K)$, hence the valuation formula can be modified:

$$\frac{c(S(0),K)}{K} = c(\frac{S(0)}{K}, 1) \quad and \quad \frac{P(S(0),K)}{K} = P(\frac{S(0)}{K}, 1)$$

The alternative representation above reduces the number of parameters needed for simulation to 4, because instead of simulate both S and K, moneyness ($\frac{S(0)}{K}$) is only simulated. The input matrix $\mathbf{X}$ is different combinations of the 4 parameters within the parameter range (table 5.1), where it is assumed the number of trading days are 252 in a year. The other parameter ranges are moneyness between 0.8 and 1.2, risk free rate of return between 1-3 % and volatility between 0.05 and 0.5.

TABLE 5.1: Parameter ranges for american put and european call options

Derivative	**r**	**T**	**Moneyness**	σ
Euro. Call	1%-3%	1d - 3y	0.8-1.2	0.05-0.5
Amer. Put	1%-3%	1d - 3y	0.8-1.2	0.05-0.5

The european put minimum and american put minimum option for two underlyings will require additional parameters, because now we have two spots, two volatilities and correlation between the assets. The first order homegeneity does not hold for multivariate contingent claims, hence the strike and the two spots are also needed. The parameters considered for the bivariate contingent claims are T, r, K, $S_1(0)$, $S_2(0)$, σ_1, σ_2 and ρ so a total of 8 parameters, and the given parameters ranges are given in table 5.2.

TABLE 5.2: Parameter ranges for exotic european put and american put options

Derivative	**r**	**T**	K	$S_1(0)$	$S_2(0)$	σ_1	σ_2	ρ
Euro. Put Min.	1%-3%	1d - 3y	80-120	100	100	0.05-0.5	0.05-0.5	0.05-0.5
Amer. Put Min.	1%-3%	1d - 3y	80-120	100	100	0.05-0.5	0.05-0.5	0.05-0.5

The simulationen of the parameter ranges is done by quasi-random Halton sequences sampling instead of e.g. random sampling like uniform sampling to obtain lower discrepancy. The Halton sequence covers the space more evenly quicker, because of the lower discrepancy. Like uniform sampling the halton sequence points is between 0 and 1, hence we need to apply a transformation to get the parameter ranges:

$$r \cdot (range\ of\ parameter) + lowerBound \quad where\ r = halton\ point$$

After the simulation of combinations of the input parameters within the given ranges the labels can be generated from existing methods. For the european call option the generation of labels is done by the classical B-S formula for call options given in proposition 3.2.1. The B-S pricing formula is well known and has an analytical solution, hence it is relatively fast to generate labels in this model. In the B-S setup in section 2.3 we assume constant volatility, which is not realistic in terms of the

known fact that volatility skew for equity options (p. 458 (Hull, 2017)). The european put minimum option is by the change of numeraire method given in closed form in section 3.3, hence the generation of labels like in the B-S price formula is fast. The american options require numerical methods, hence the generation of labels are more computational expensive. The labels for the american options are generated by CRR for american put option with 1 underlying stock and by BEG for two underlyings presented in chapter 3. When the datasets ($\mathbf{X}$, $\boldsymbol{y}$) are generated we are left with a regression task. The regression task is to predict the price $\boldsymbol{y}$ from the input parameters $\mathbf{X}$.

The datasets generated for training the model will be the in-sample data set, where the parameters are sampled within the parameter ranges for the given derivative. The in-sample dataset are split up into a training and validation dataset in order to measure model performance. The training dataset is used to update the parameters weights,biases, etc. internal in the model and finetune hyperparameters for choosing the optimal model design. To avoid overfitting and good generalization models the validation set is useful, because the trained model has not seen the data before evaluation on validation set. The validation set is randomly subsampled from the training set, where the validation dataset constitute 20 procent of the training set. To check the robustness of the regression we choose to sample two test datasets out-of-sample, which means we simulate data with modified parameter ranges.

The test datasets out-of-sample is simulated by adjusting the parameter range for either moneyness or maturity for the options with 1 underlying. For the options with two underlyings the parameter ranges for the spot and the maturity is modified for the out-of-sample test sets. The test dataset has not been seen by the model in the training process, hence we get a unbiased evaluation of the model. The aim with producing different test and validation data sets is to measure the models performance at interpolation and extrapolation. We have illustrated the marginal distributions for the american put option with a training dataset of 300.000 samples. The marginal distributions (figure 5.1) shows that we have succesfully generated parameters in the given ranges and the parameters are evenly spaced in the ranges.

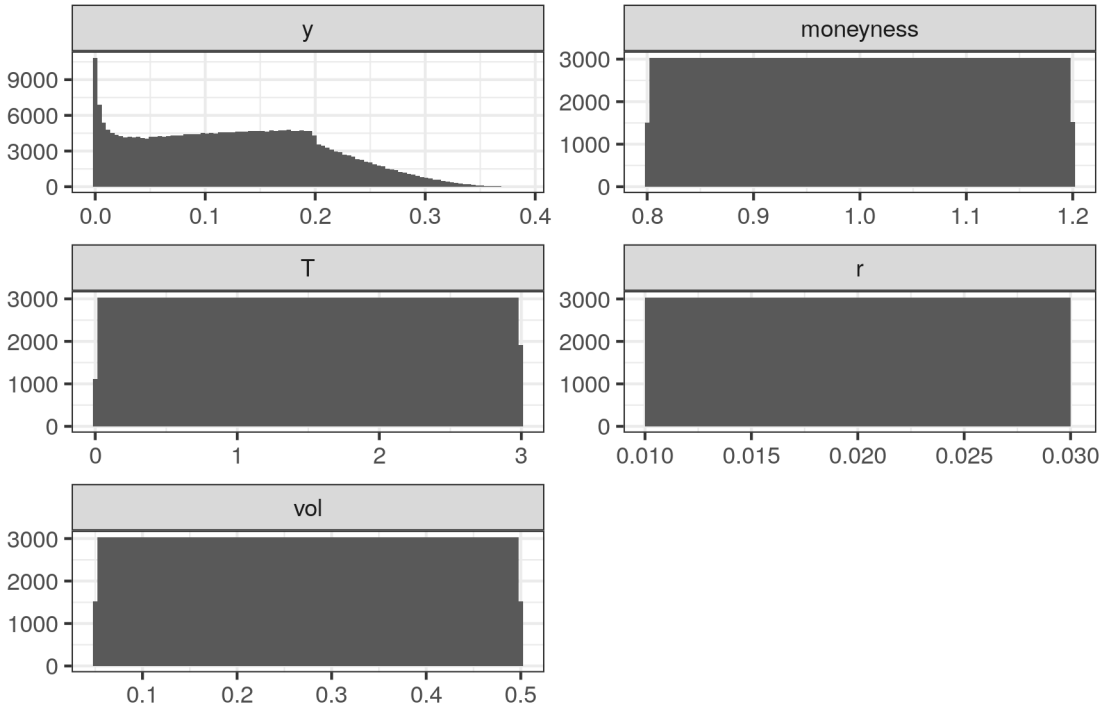

FIGURE 5.1: Quasi random simulation with halton sequence for input variables and CRR for generation of labels for american put option.

The marginal distributions shown is for 300.000 data samples $(\mathbf{X}, y)$ generated by halton sequences and CRR model, where the marginal distributions for the features cover the parameter range almost uniform and the simulated y lies with most values at zero and maximum at 0.387 rounded to three decimals. In the model performance section out-of-sample dataset will be used to check the predictive strength of the models. The out-of-sample test dataset will be 60.000 generated with halton sequences sampling like the in-sample dataset and the parameter ranges for the options with 1 underlying is given in 5.3.

TABLE 5.3: Parameter ranges for european call and american put option

Dataset	Derivative	**Moneyness**	**r**	σ	**T**
In-Sample	Euro. Call	0.8-1.2	1%-3%	0.05-0.5	1d-3y
Out-Of-Money		0.6-0.8	1%-3%	0.05-0.5	1d-3y
Longer Maturity		0.8-1.2	1%-3%	0.05-0.5	3y-5y
In-Sample	Amer. Put	0.8-1.2	1%-3%	0.05-0.5	1d-3y
Out-Of-Money		1.2-1.4	1%-3%	0.05-0.5	1d-3y
Longer Maturity		0.8-1.2	1%-3%	0.05-0.5	3y-5y

The out-of-sample for the multivariate contigent claims will be similar, except that the strike will be between 120-140, because first order homegeneity does not hold for the multivariate contingent claims.

5.2.2 Training

The in-sample data is the simulated data within the parameter ranges for the different types of derivatives, where the aim is that the model will learn the pricing function f^*. Once the dataset $(\mathbf{X}, \boldsymbol{y})$ is generated the model can be trained to approximate the true function f^*. To train the model we use MLPs regression to infer the pricing function from the generated data, hence the approach is model free, because the data is only needed. We will also present a standard linear model to compare the MLPs with the classical regression methods. We want to have a fast, robust and accurate model after training on the training set. To train the model we need a measure for the error, where the standard mean squared error (MSE) for regression is applied. I.e. the cost function chosen to be the empirical risk function with a quadratic loss function:

$$J(\theta) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

The MSE penalizes outliers stronger than e.g. mean absolute error (MAE), but it penalize small deviations less. To update the weights the Adam optimization algorithm is chosen with learning rate set to $\eta = 0.001$, which is a standard choice for MLPs.

We test empirically the best hyperparameters for the MLPs, where the validation set is used. The first investigation is to look how well the model performs on varying dataset sizes, where the goal is to quantify how large datasets is needed for having a high quality model. The datasets for an european call option considered are in-sample datasets of size 100, 1K, 10K, 100K, 300k, 1M data samples, where the validation set is subsampled from the training data set. By inspiration from (Hirsa, Karatas, and Oskoui, 2019) we choose to test the datasets with a MLPs with 4 layers, 120 neurons in each hidden layer and 1 output. In each layer we choose the activation function leaky ReLU, which is one of the most popular choices for activation functions. The number of data samples is relevant for real data, because for real market data there is not unlimited market data available.

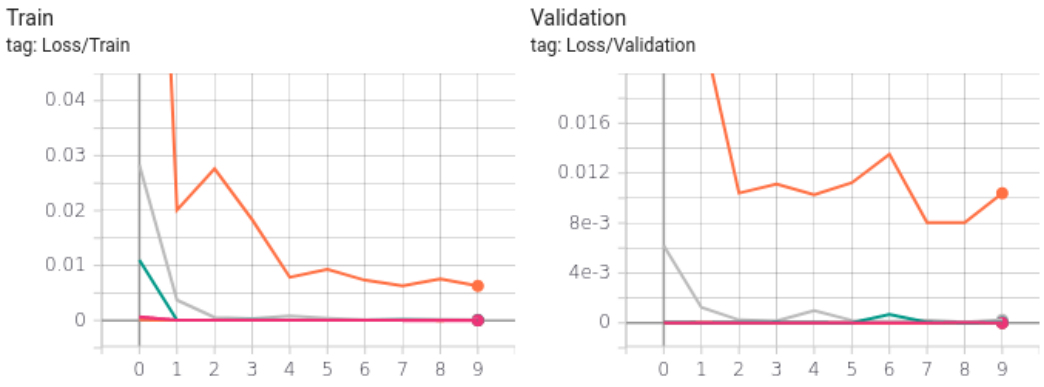

FIGURE 5.2: Effect of gather more data on training and validation loss, where the x-axis is number of epoch-1 and y-axis is the loss for the training and validation loop. The orange line with the largest is for 100 datapoints, the grey line for 1000 datapoints, the green line 10K datapoints, the three last lines blends together where it is for 100K,300K and 1M datapoints. More can be seem at [TensorBoard EuroMLP]

From figure 5.2 we see that the the model performs very well for in-sample data with only 1K datapoints, when looking closer at the tensorboard the validation error is not lower for 1M datapoints than 100K and 300K datapoints. The model is only trained once on each datasets, so there is some randomness on each run, but the picture is clear. The model to interpolate prices for european call options in-sample data does not significant improve with gathering more data than 1K-10K datapoints this is good news for using the method on real market data. In the article (Gaspar, Lopes, and Sequeira, 2020) they conduct a study to price american put options with real market data, where they conclude they got better results than for LSM with around 40K datapoints which is in line with the result in figure 5.2. Be aware that the simulated data can underestimate the validation error, because we have a controlled setup where the parameter range is within a predetermined range. For our controlled setup with simulated data we can choose arbitrary many datapoints albeit making the method more computational expensive. By above discussion we choose to work with 300K datapoints i.e. the same dataset size as in (Hirsa, Karatas, and Oskoui, 2019) in order to compare results, the new thing compared to (Hirsa, Karatas, and Oskoui, 2019) is that we will also try to price bivariate contingent claims with the MLPs regression method. The main focus in the rest of this section is to train the model for pricing multivariate contigent claims.

TABLE 5.4: Performance of in-sample and out-of-sample dataset of different sizes for MLPs regression on a european call option

Model	**Dataset**	**MSE**	**RMSE**	**MAE**	R^2
MLPs	In-sample Training: 800	0.000092	0.009601	0.007214	0.989676
MLPs	8K	0.000015	0.003875	0.002999	0.998518
MLPs	80K	0.000008	0.002914	0.002466	0.999167
MLPs	240K	0.000006	0.002419	0.001966	0.99942
MLPs	800K	0.000004	0.002098	0.001742	0.999565
MLPs	In-sample Validation: 200	0.000092	0.009601	0.007214	0.989676
MLPs	2K	0.000015	0.003875	0.002999	0.998518
MLPs	20K	0.000008	0.002914	0.002466	0.999167
MLPs	60K	0.000006	0.002419	0.001966	0.99942
MLPs	200K	0.000004	0.002098	0.001742	0.999565

The standard measures mean square error (MSE) and root mean square error (RMSE) shows that the MLPs increase its precision for in-sample interpolation, but for the out-of-sample datasets the decreasing error hits a plateau for 80K data samples. This

and out-of-sample datasets with 60K samples

actual data samples agrees with the model for both in-sample and out-of-sample. The in-sample training is based on the parameter ranges in table 5.6, where the out-of-sample is 60.000 simulated samples, where the moneyness lies between 0.6-0.8 and the other ranges remain the same. Table 5.4 shows how the validation error for in-sample and the test-error for out-of-sample datasets perform varying the number size of the training dataset. The validation error declines for increasing sample size, but the test error does not improve significantly for 100K up to 1M. The test error decreases for 100K-300K, but it increases for 300K-1M, hence we choose to work with the 300K dataset in the analysis. The RMSE match units - standard deviation

of residuals how much spread away from mean Coefficient of determinant metric of correlation variation around the mean. predictic how the regression does compared to only fitting with the mean Does our regression fit better the data than the mean less variation around the mean. Hence most variation is explained by the regression. some number less variation around the line vs the mean agrees fits for most samples, but for

The same architecture is applied to each dataset and the Adam optimizer uses minibatches of size 64.

The training algorithm is

Training the model is how the mode does not know anything about the function , where . The architecture of the network where in hyperparameter tuning, we will try with elu in each layer instead. The architecture is chosen based on the investigation done in (Hirsa, Karatas, and Oskoui, 2019), where we in the next subsection try to see how many samples is needed for good performance. A research area within deep learning is to tune the hyperparameters to the specific task, where both manually search and the automated grid search are used.

5.2.2.1 Hyperparameter Tuning

The hyperparameter chosen is based on (Hirsa, Karatas, and Oskoui, 2019), nevertheless we will try with elu. ELU has linear asymptotes, which is the desired behaviour in valuation problems: financial products are often linearly extrapolated and this behaviour is often enforced. For example, in finite difference methods, we generally work with linear boundary conditions. ELU is also considered a best practice presently in the deep learning community, for very different reasons related to speed and stability of numerical optimization.

5.2.2.2 Polynomial Regression

The dataset in the MLPs regression is the same for the polynomial regression, further we choose same performance metrics, but the model and model training is obvously different from the MLPs. There is exists now a closed form solution for the optimation problem by solving the "normal equations" and we have a linear model. We fit polynomials up to degree 6 for comparision of the model capacity and fit.

$$y_i = \beta_0 + \beta_1 \cdot x_i + \cdots + \beta_n \cdot x_i^n + \epsilon_i \quad \textit{where } n = 1, 2, \ldots, 6$$

From the illustration (figure 5.3) is it clear, that the in-sample fit improves with increased model capacity. The linear regression is too simple for pricing european option, but it looks like the 6 order polynomial actually performs better than the MLPs in the in-sample test set (see also table 5.5). It is important to note that we want predictive strength for our model, i.e. a small generalization. The out-of-sample data will reveal if the high order polynomial or MLPs have overfitted the data.

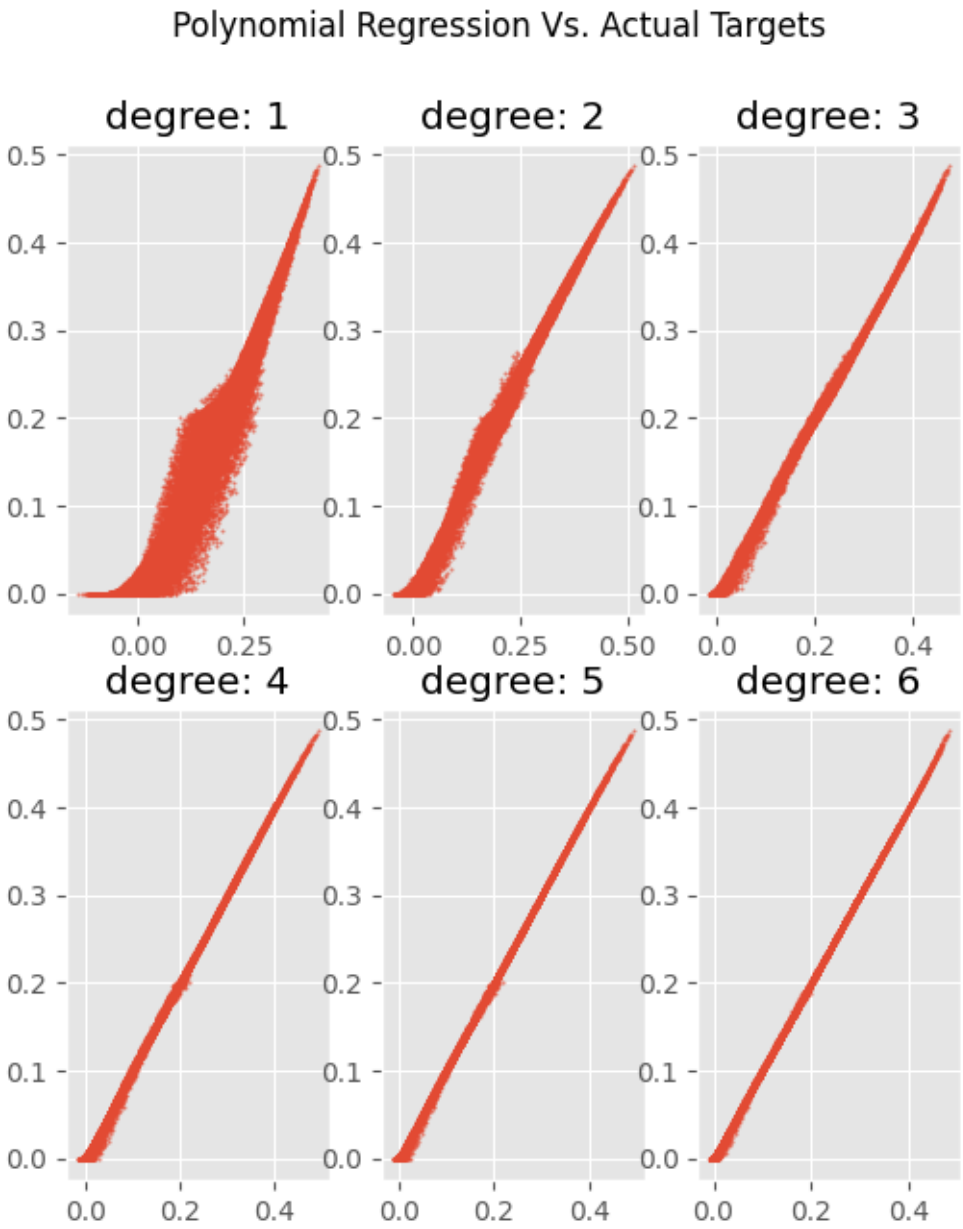

FIGURE 5.3: Predicted price based on polynomial regression of varying degree

The table is created to compare the performance for each model. The table confirms that the linear regression has a worse fit than the other models with higher capacity. The difference on the MLPs and higher order polynomial regression models less than $6 \cdot 10^{-4}$ for coefficient of determination and less than $15 \cdot 10^{-3}$. The difference is negible so the fit for MLPs and polynomial regression of degree 4-6 performs all very well on the data.

TABLE 5.5: Prediction results for european call test data for in sample polynomial regression

Model	Dataset	MSE	RMSE	MAE	R^2
Linear Reg.	In-Sample	0.000631	0.025122	0.018264	0.937445
2. degree		0.000069	0.008298	0.006136	0.993175
4. degree		0.000004	0.002059	0.001282	0.999580
5. degree		0.000002	0.001407	0.000864	0.999804
MLPs		0.000003	0.001628	0.001337	0.999736
3. degree	In-Sample	1.31e-05	0.00362470	0.002559	0.998698
6. degree	In-Sample	9.23e-07	0.000961	0.000592	0.999908
MLPs	In-Sample	0.000006	0.002419	0.001966	0.99942

5.2.3 Model Performance In-Sample

The model performance is evaluated by MSE, RMSE, MAE and coefficient of determination, where all the measures evaluate how close is the model predictions with the actual targets. For a high quality model the first three measures should be close to 0, where the latter should be close to 1. For MSE close to 0 means that the model predictions does not differ a lot from the observed targets. The RMSE and MAE are same kind of measure, but just measured slightly different. The RMSE is the square root of MSE, which means MSE and RMSE penalize large errors. The MAE is the mean absolute error and large errors is penalized less. Coefficient of determination provides a measure of how well observed targets are replicated by the model, based on the proportion of total variation of target explained by the model.

TABLE 5.6: Prediction results for european call test data for in sample

MSE	RMSE	MAE	Coefficent of Determination
0.000006	0.002419	0.0019661242	0.99942

The performance measures are generally good, we see MAE, MSE and RMSE all have values less than 0.002419 from zero and a coefficient of determination 0.00058 from 1.

Illustration of the model fit is also provided, where the plot shows $\frac{c(S_0,K)}{K}$ predicted from the model and observed target values. The conclusion from the performance metrics is also present in the figure, where we see the model predicts close to target values over the whole range. Before moving on to pricing for american options, we investigate if polynomial regression can perform as MLPs

5.2.4 Out of sample predictions

All the models considered had good performance on the in-sample dataset except of the too simple models linear regression and 2. degree polynomial regreesion. To test the predictive strength for the models, we test the models with out-of-sample data. Specifically we consider longer maturity and deep-out-of-money options (see table 5.7)

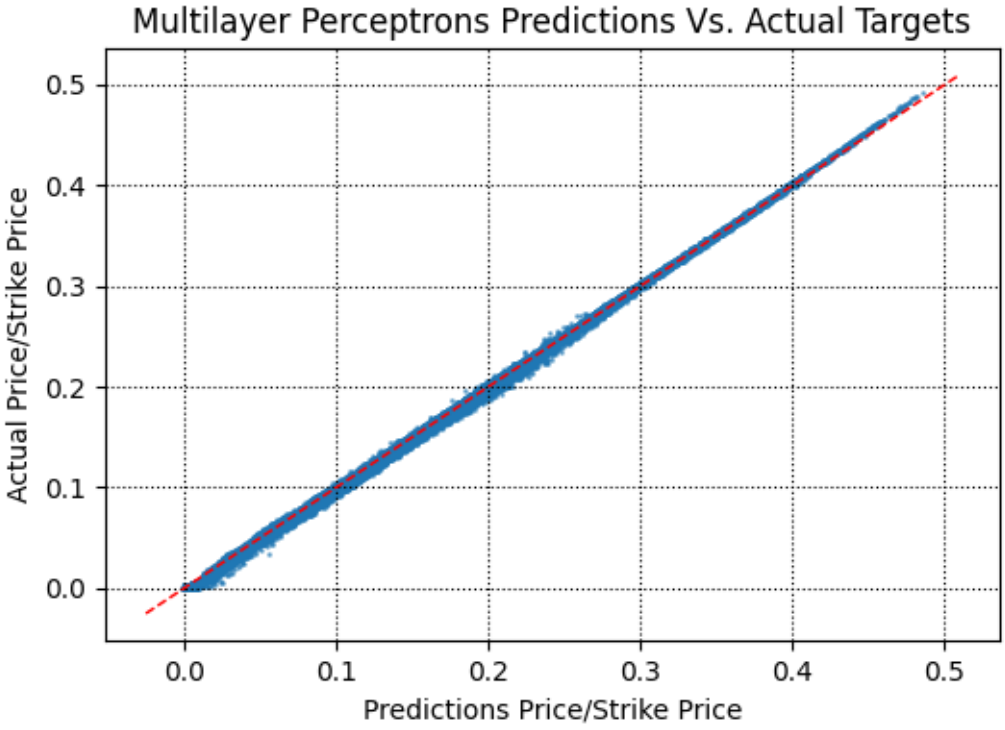

FIGURE 5.4: Predicted price based on MLPs model

TABLE 5.7: Parameter range

Dataset	**Moneyness**	**r**	σ	**T**
In-Sample	0.8-1.2	1%-3%	0.05-0.5	1/252-3.0
Out-Of-Money	0.6-0.8	1%-3%	0.05-0.5	1/252-3.0
Longer Maturity	0.8-1.2	1%-3%	0.05-0.5	3-0-5.0

The performance measures show that the polynomial regression that was performing better on the In-Sample dataset was due to overfitting, because the high order polynomial regression does perform poorly on out-of-sample data (see table 5.9 and figure 5.5). For the 5. order polynomial regression, we see a negative coefficient of determination, which means the model performs worse than the model with the mean as a horizontal line. This means the 5. order polynomial clearly have low predictive strength.

TABLE 5.8: Performance of predictive strength for different regression models

Model	Dataset	MSE	RMSE	MAE	R^2
Linear Reg.	In-Sample	0.000631	0.025122	0.018264	0.937445
2. degree		0.000069	0.008298	0.006136	0.993175
4. degree		0.000004	0.002059	0.001282	0.999580
5. degree		0.000002	0.001407	0.000864	0.999804
MLPs		0.000003	0.001628	0.001337	0.999736
Linear Reg.	Out-Of-Money	0.005772	0.075973	0.060936	-2.377251
2. degree		0.000767	0.027694	0.022203	0.551246
4. degree		0.000944	0.030724	0.020542	0.447668
5. degree		0.001812	0.042568	0.027125	-0.060261
MLPs		0.000005	0.002152	0.001569	0.997291
Linear Reg.	Longer Maturity	0.002662	0.051593	0.041232	0.818143
2. degree		0.001196	0.034577	0.026287	0.918316
4. degree		0.003956	0.062894	0.039932	0.729744
5. degree		0.012255	0.110702	0.064402	0.162742
MLPs		0.000066	0.008138	0.005817	0.995475

We see that the 2. degree polynomial is best on most of the performance measures for our-of-sample data. Notice based on MAE for out-of-money data the 4. degree polynomial performs better than the 2. degree polynomial, but if we look at the MSE the performance is best for the 2. degree polynomial. This is due to that the two measure weight errors in different ways, where the MSE penalize large error more than the MAE. Amoung the polynomial the 2. degree polynoial regression performs best, but gets outperformed by MLPs (see also figure 5.6). The MLPs has high predictive strength compared to the polynomials, because it performs well also on out-of-sample dataset. The section showed how succesfully MLPs can be used in a GBM model setup for pricing european options. The MLPs is not based on a model, it can only see data. This is promising because the MLPs could also be used for actual market data or different models to learn patterns. In the coming sections we will focus on american put options and basket options, where in the basket option case the MLPs benefit for the fact that it does not increase expontially with the dimension to maintain low error compared to polynomial regression. For the american put option the MLPs regression will only be considered, because the polynomial regression have low predictive strength for the simpler european option.

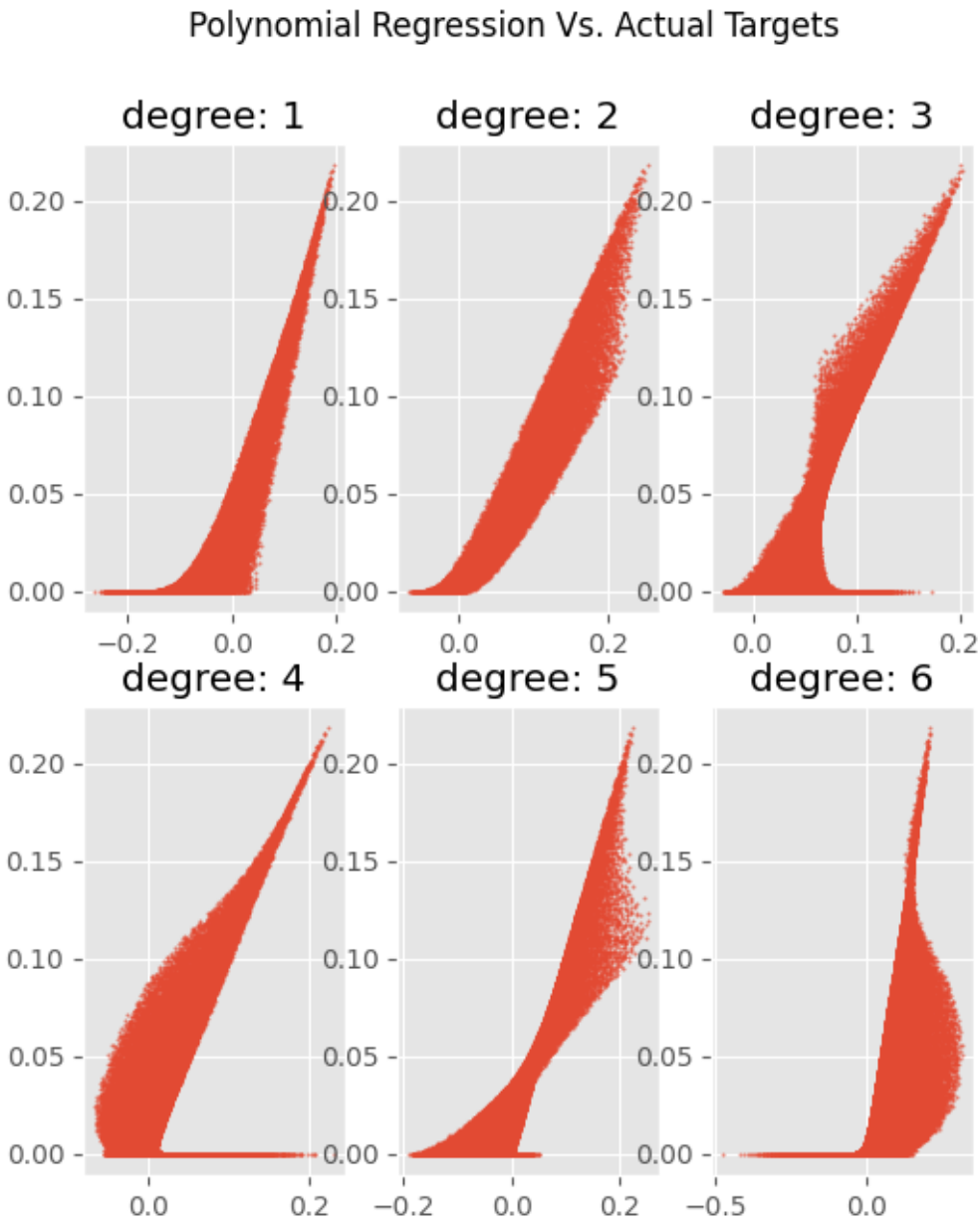

FIGURE 5.5: Predicted price based on polynomial regression of varying degree

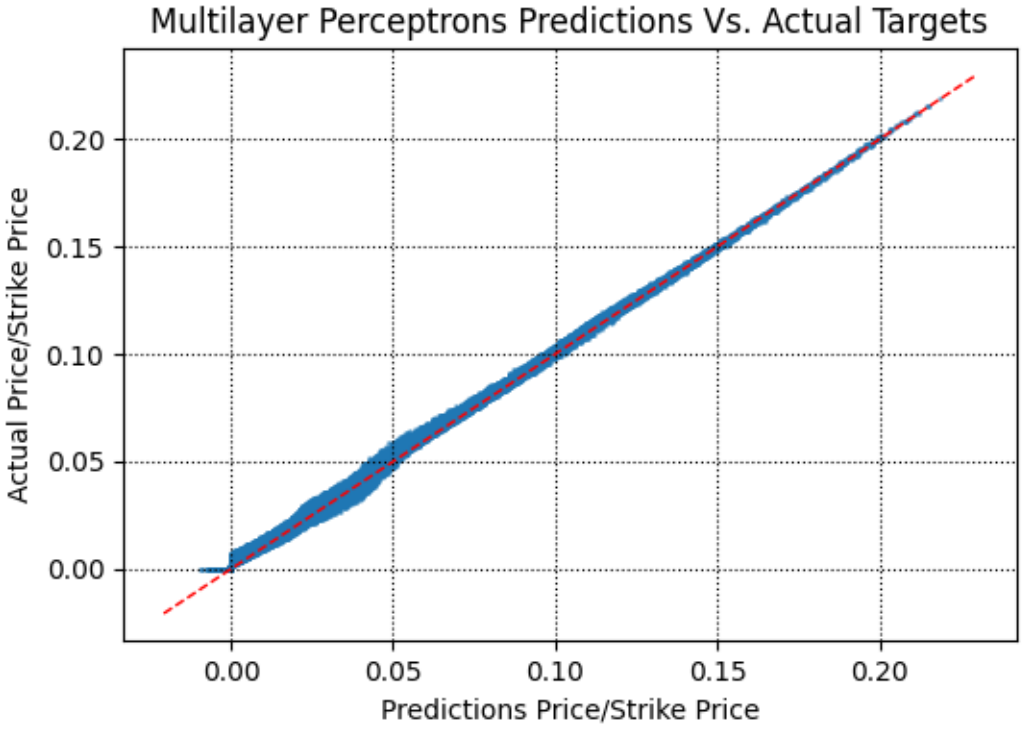

FIGURE 5.6: Predicted price based on MLPs model

5.3 Multilayer Perceptrons Regression For American Options

5.3.1 Data

5.3.2 Optimization and cost function

5.3.3 Model Performance

TABLE 5.9: Performance of predictive strength for different regression models

Model	Dataset	MSE	RMSE	MAE	R^2
MLPs Reg.	In-Sample	0.000002	0.001562	0.001278	0.999634
MLPs Reg.	Out-Of-Money	0.000030	0.005503	0.003925	0.989674
MLPs Reg.	Longer Maturity	0.000194	0.013922	0.010731	0.980759

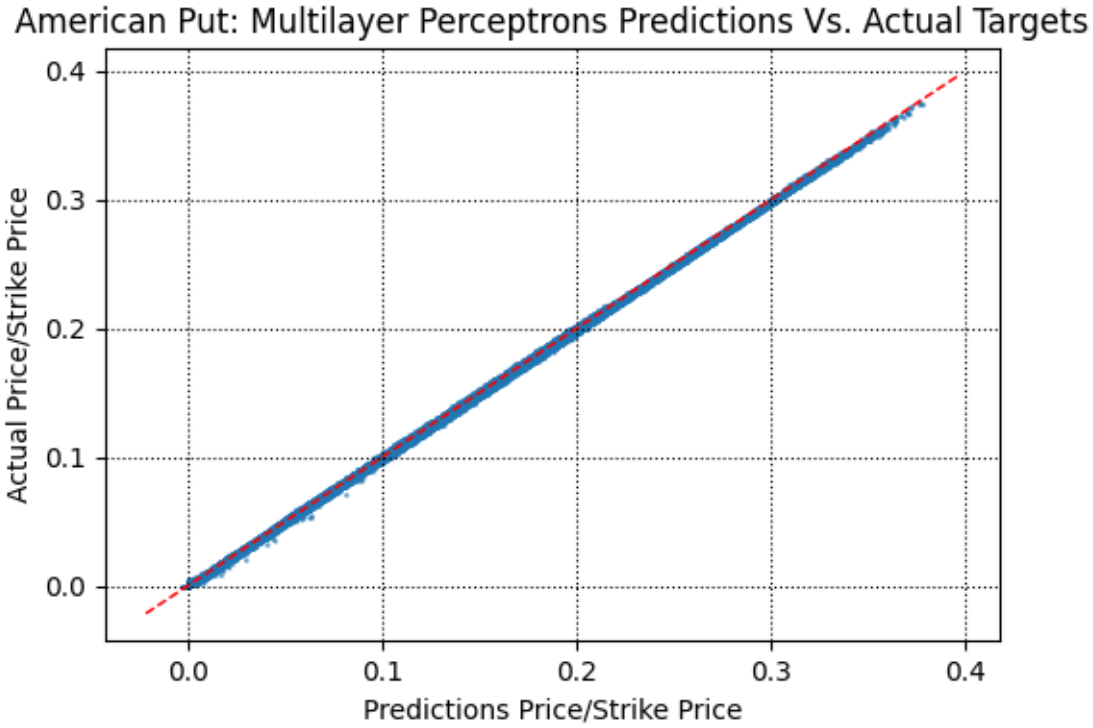

FIGURE 5.7: Predicted price based on MLPs model, where the targets are from the binomial model

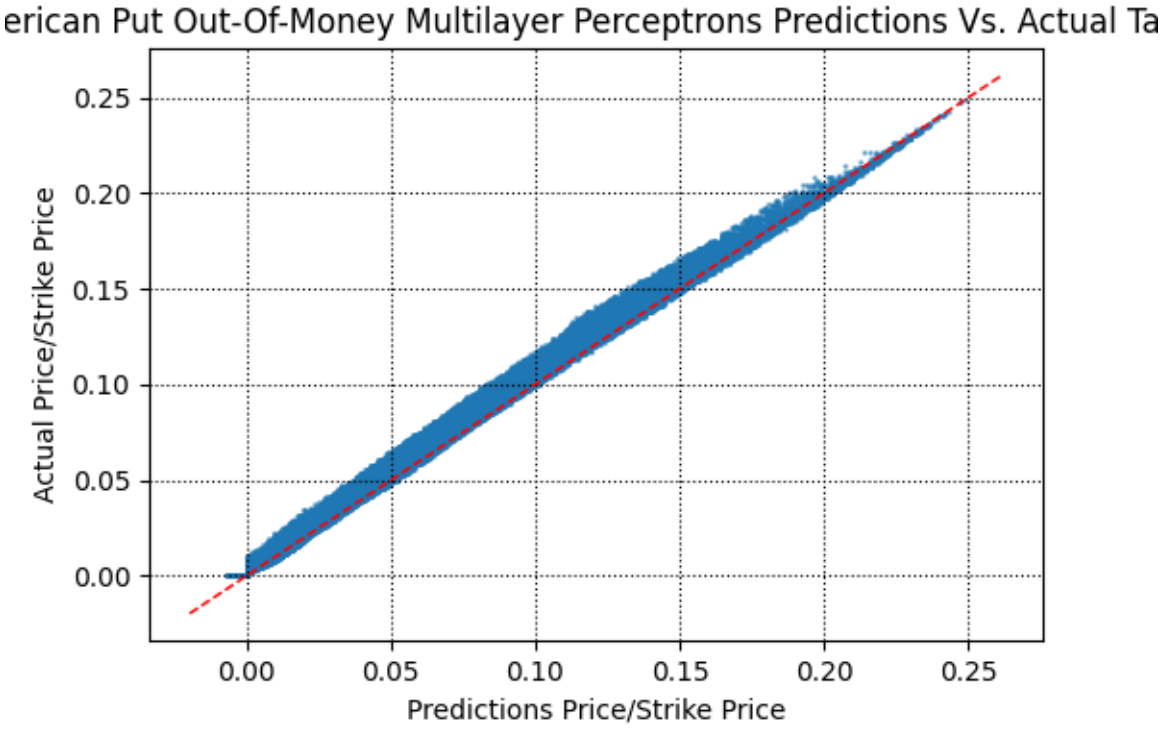

FIGURE 5.8: Predicted price based on MLPs model, where the targets are from the binomial model

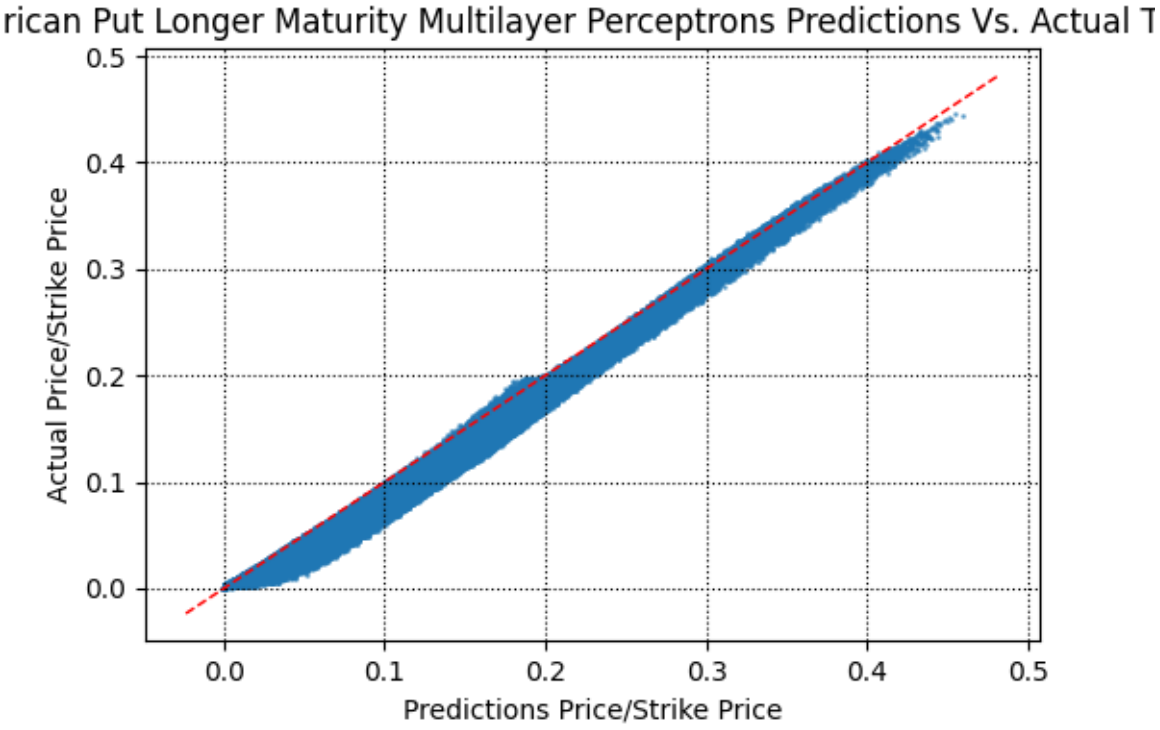

FIGURE 5.9: Predicted price based on MLPs model, where the targets are from the binomial model

To compare the results by

The results are promosing, because the MLPs are model free in the sense, that we could have simulated out for any model, where MLPs then learn from data only. By observing this the method can be extended to real market data in fact (Gaspar, Lopes, and Sequeira, 2020) have resently showed good results for a MLPs for real market data. The next section will investigate the curse of dimensionality and the application of MLPs in this context. (Ferguson and Green, 2018)

TABLE 5.10: Valuation of American put option with K=40 and r=0.06.

Spot	σ	**T**	**MLPs Regression**	**Binomial Tree**	**LSM**	**abs. diff.**
36	0.2	1	4.584	4.488	4.478	0.010
36	0.2	2	4.649	4.846	4.828	0.018
36	0.4	1	7.090	7.119	7.092	0.027
36	0.4	2	8.487	8.508	8.500	0.008
38	0.2	1	3.094	3.260	3.245	0.015
38	0.2	2	3.638	3.748	3.735	0.013
38	0.4	1	6.172	6.165	6.144	0.021
38	0.4	2	7.605	7.689	7.665	0.024
40	0.2	1	2.114	2.316	2.313	0.003
40	0.2	2	2.779	2.885	2.881	0.004
40	0.4	1	5.274	5.310	5.326	0.016
40	0.4	2	6.839	6.914	6.908	0.006
42	0.2	1	1.494	1.622	1.622	0.000
42	0.2	2	2.167	2.217	2.212	0.005
42	0.4	1	4.548	4.602	4.596	0.006
42	0.4	2	6.197	6.264	6.243	0.021
44	0.2	1	1.000	1.117	1.113	0.004
44	0.2	2	1.678	1.697	1.688	0.009
44	0.4	1	3.949	3.956	3.962	0.006
44	0.4	2	5.649	5.656	5.649	0.007

Chapter 6

Discussion and Further Investigation

6.1 Main Section 1

6.1.1 Subsection 1

6.1.2 Subsection 2

6.2 Main Section 2

Chapter 7

Conclusion

7.1 Main Section 1

7.1.1 Subsection 1

Appendix A

Mathematical results and definitions

Theorem A.0.1. ***Itô's formula multidimensional*** *Let the n-dimensional process X have dynamics given by:*

$$dX(t) = \mu(t)dt + \sigma(t)dW(t) \tag{A.1}$$

Then the process $f(t, X(t))$ has stochastic differential given by:

$$df(t,X(t)) = \frac{\partial f(t,X(t))}{\partial t}dt + \sum_{i=1}^{n}\frac{\partial f(t,X(t))}{\partial x_i}dX_i(t) + \frac{1}{2}\sum_{i,j=1}^{n}\frac{\partial^2 f(t,X(t))}{\partial x_i \partial x_j}dX_i(t)dX_j(t) \tag{A.2}$$

Note:

$$dW_i \cdot dW_j = \begin{cases} \rho_{ij}dt & \textbf{\textit{For correlated Weiner processes}} \\ 0 & \textbf{\textit{For independent Weiner processes}} \end{cases}$$

(see page 58-60 (Björk, 2009))

Theorem A.0.2. ***The Girsanov Theorem*** *Assume the probability space $(\Omega, \mathcal{F}, P, \mathcal{F}_t^{W^p})$ and let the Girsanov kernel ϕ be any d-dimensional adapted column vector process. Choose a fixed T and define the process L on $[0, T]$ by:*
$dL_t = \phi(t)^T \cdot L_t d\bar{W}_t^P$
$L_0 = 1$.
Assume that $E^P[L_T] = 1$ and define the new probability measure Q on $\mathcal{F}_T$ by:

$$L_T = \frac{dQ}{dP} \quad on\ \mathcal{F}_T$$

Then

$$d\bar{W}(t) = \phi(t)dt + dW(t) \tag{A.3}$$

Where $W(t)$ is the Q-Wiener process and $\bar{W}(t)$ is the P-Wiener process (see page 164 (Björk, 2009))

Definition A.0.1. Stopping time in continuous time: A nonnegative random variable τ is called a stopping time w.r.t. the filtration $\mathcal{F}$ if it satisfies the condition:

$$\{\tau \leq t\} \in \mathcal{F}_t \quad \forall t \geq 0 \tag{A.4}$$

(see page 329 (Björk, 2009))

Definition A.0.2. Orthogonal vectors: Two vectors $\overrightarrow{a}$ and $\overrightarrow{b}$ are orthogonal, if their dot product is 0:

$$\overrightarrow{a} \cdot \overrightarrow{b} = 0$$

We will use the notation:

$$\overrightarrow{a} \perp \overrightarrow{b} \tag{A.5}$$

Bibliography

Björk, Thomas (2009). *Arbitrage Theory in Continuous Time*. Third edition. Oxford.

Black, Fischer and Myron Scholes (1973). "The Pricing of Options and Corporate Liabilities". In: *The Journal of Political Economy* 81.3, pp. 637–654. URL: http://www.jstor.org/stable/1831029 (visited on 02/09/2020).

Boyle, Phelim P., Jeremy Evnine, and Stephen Gibbs (1989). "Numerical Evaluation of Multivariate Contingent Claims". eng. In: *The Review of financial studies* 2.2, pp. 241–250. ISSN: 0893-9454.

Buffett, Warren (2002). "Berkshire Hathaway's (BRK.B) annual letters to shareholders". In: URL: https://www.berkshirehathaway.com/letters/2002pdf.pdf (visited on 06/23/2020).

— (2008). "Berkshire Hathaway's (BRK.B) annual letters to shareholders". In: URL: https://www.berkshirehathaway.com/letters/2008ltr.pdf (visited on 06/23/2020).

Cox, John and Mark Rubinstein Stephen Ross (1979). "Option princing: A simplified approach". In: *Journal of Financial Economics* 7, pp. 229–263.

Ekvall, Niklas (1996). "A lattice approach for pricing of multivariate contingent claims". In: *European Journal of Operational Research* 91.2, pp. 214–228. URL: https://EconPapers.repec.org/RePEc:eee:ejores:v:91:y:1996:i:2:p:214-228.

Ferguson, Ryan and Andrew Green (2018). "Deeply Learning Derivatives". eng. In:

Gaspar, Raquel M, Sara D Lopes, and Bernardo Sequeira (2020). "Neural Network Pricing of American Put Options". eng. In: *Risks (Basel)* 8.3, pp. 73–. ISSN: 2227-9091.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville (2016). *Deep Learning*. http://www.deeplearningbook.org. MIT Press.

Hirsa, Ali, Tugce Karatas, and Amir Oskoui (2019). "Supervised Deep Neural Networks (DNNs) for Pricing/Calibration of Vanilla/Exotic Options Under Various Different Processes". eng. In:

Hull, John C. (2017). *Options, Futures, and Other Derivatives*. Vol. Ninth edition. Pearson Global Education.

Johnson, Herb (1987). "Options on the Maximum or the Minimum of Several Assets". In: *The Journal of Financial and Quantitative Analysis* 22.3, pp. 277–283. URL: https://www.jstor.org/stable/2330963?seq=1.

Kohler, Michael, Adam Krzyżak, and Nebojsa Todorovic (2010). "PRICING OF HIGH-DIMENSIONAL AMERICAN OPTIONS BY NEURAL NETWORKS". eng. In: *Mathematical finance* 20.3, pp. 383–410. ISSN: 0960-1627.

Longstaff, Francis A. and Eduardo S. Schwartz (2001). "Valuing American Options by Simulation: A Simple Least-Squares Approach". In: *The Review of Financial Studies*.

MacKay, David J. C. (2018). *Information theory, inference and learning algorithms*. eng. Repr. with corr. Cambridge University Press. ISBN: 0521642981.

Merton, Robert C. (1973). "Theory of Rational Option Pricing". In: *The Bell Journal of Economics and Management Science* 4.1, pp. 141–183. URL: http://links.jstor.

org/sici?sici=0005-8556%28197321%294%3A1%3C141%3ATOROP%3E2.0.CO%3B2-0 (visited on 05/14/2020).

Ouwehand, P. (2006). "PRICING RAINBOW OPTIONS". In:

Shiryaev, Albert and Goran Peskir (2006). *Optimal Stopping and Free-Boundary Problems*. eng. Lectures in Mathematics. ETH Zurich. Basel: Springer Basel AG. ISBN: 3764324198.